	L #	Hits	Search Text
1	L3	781	compress$3 near5 instruction
2	L4	286	compact$3 near5 instruction
3	L7	7	(expand$3 conver$4) near20 1
4	L6	23	(expand$3 conver$4) near20 4
5	L5	83	(expand$3 decopmress$3 conver$4) near20 3
6	L1	187	abbreviat$3 near5 instruction
7	L8	461	(shorter shortened) near5 instruction
8	L10	9	(transform$3 translat$3 conver$4) near20 8
9	L12	12	(transform$3 translat$3 conver$4) near20 (1 3 4) not (5 6 7)
10	L11	1	(transform$3 translat$3 conver$4) near20 9
11	L9	97	((smaller less) adj3 bit) near5 instruction
12	L13	2205	(original native) near5 instruction
13	L14	4	(1 3 4 8 9) near20 13 not (5 6 7 10 11 12)

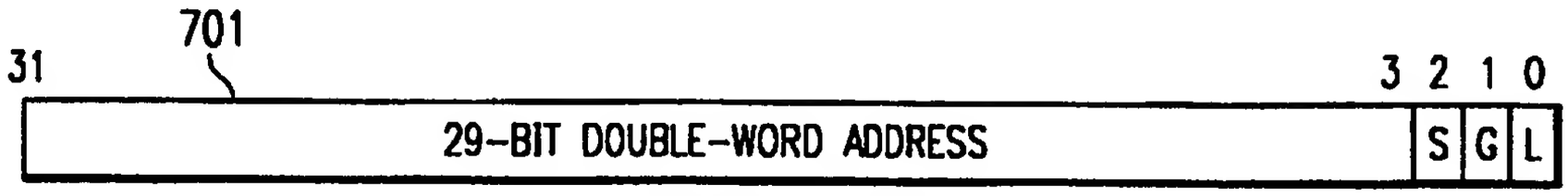


FIG. 32

FIG. 33

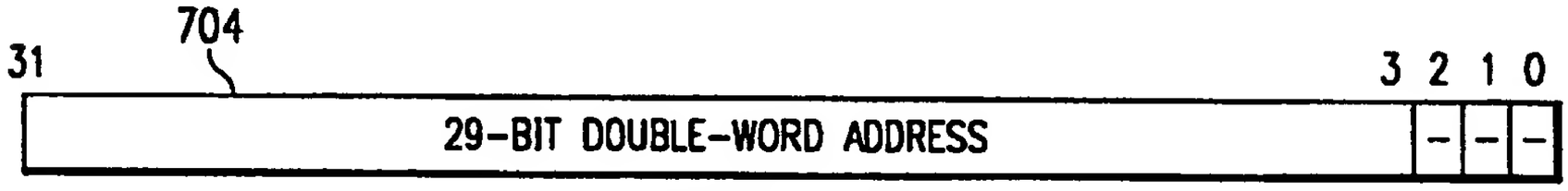


FIG. 34

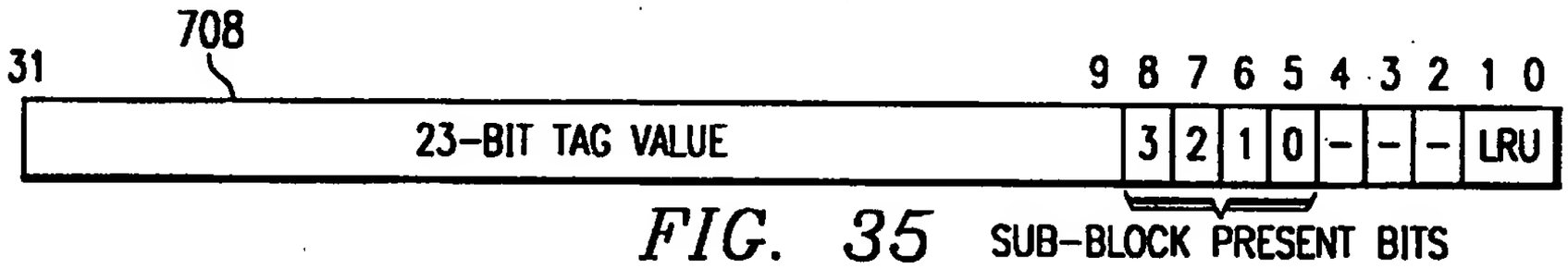


FIG. 35

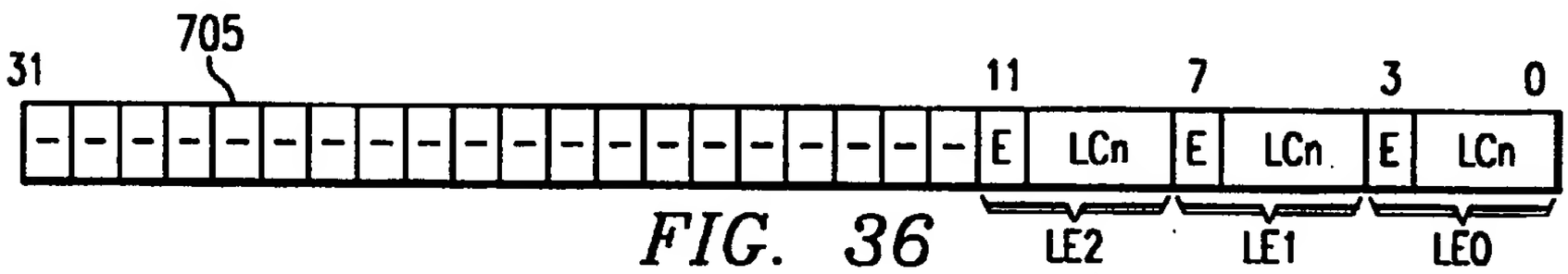


FIG. 36

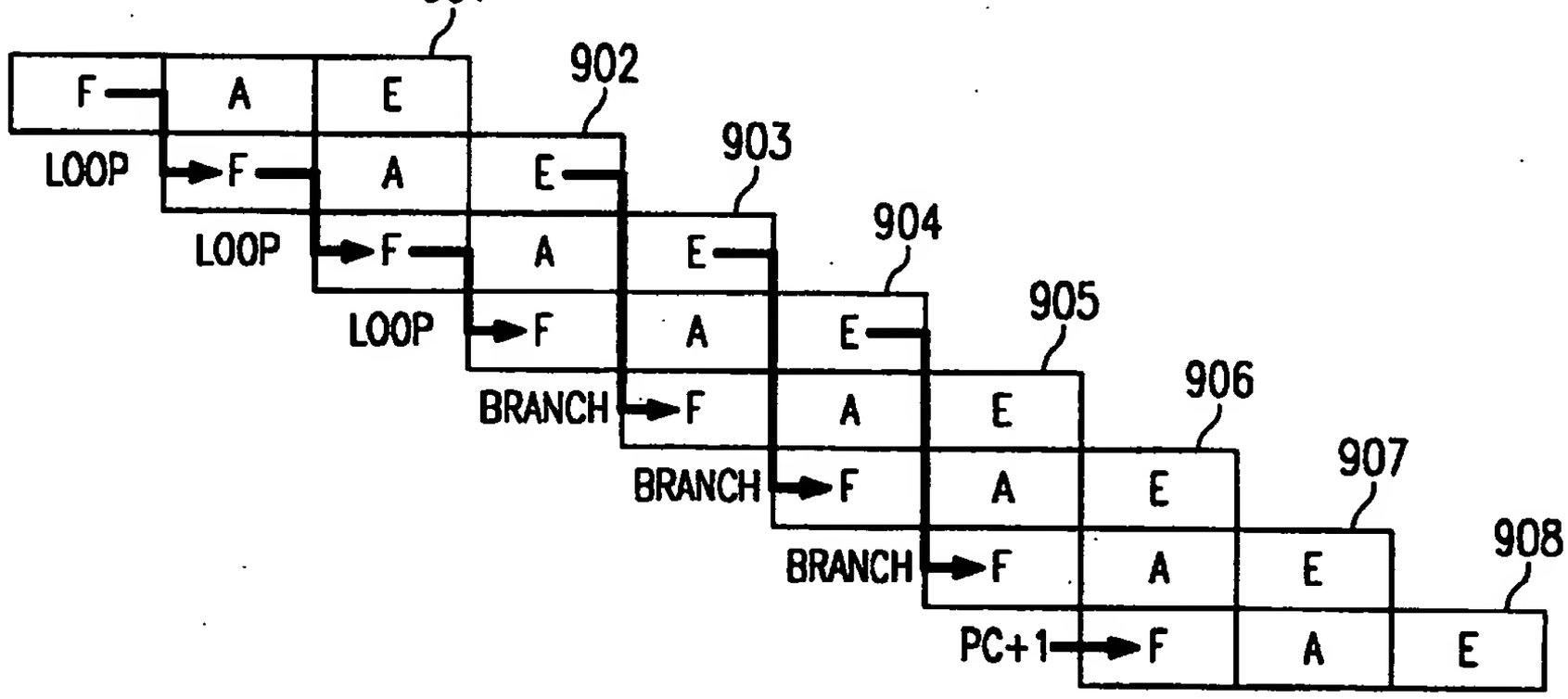


FIG. 39

	Document	U	Title	Current OR
1	USD 61015 14 A	☐	Anti-aliasing apparatus and method with automatic snap fit of horizontal and vertical edges to target grid	707/530
2	US 59737 46 A	☒	Image data conversion processing device and information processing device having the same	348/458
3	US 59076 94 A	☒	Data processing apparatus for performing a pipeline operation on a load and extension instruction	712/210
4	US 59037 60 A	☒	Method and apparatus for translating a conditional instruction compatible with a first instruction set architecture (ISA) into a conditional instruction compatible with a second ISA	717/7
5	US 58359 63 A	☒	Processor with an addressable address translation buffer operative in associative and non-associative modes	711/207
6	US 54505 60 A	☒	Pointer for use with a buffer and method of operation	711/200
7	US 50034 71 A	☒	Windowed programmable data transferring apparatus which uses a selective number of address offset registers and synchronizes memory access to buffer	710/56
8	US 44082 75 A	☒	Data processing system with data cross-block-detection feature	711/5
9	US 43171 70 A	☒	Microinstruction controlled data processing system including micro-instructions with data align control feature	712/224

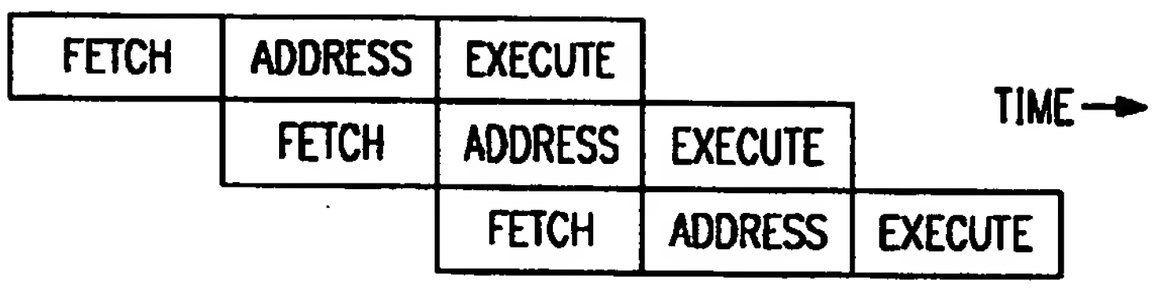


FIG. 4

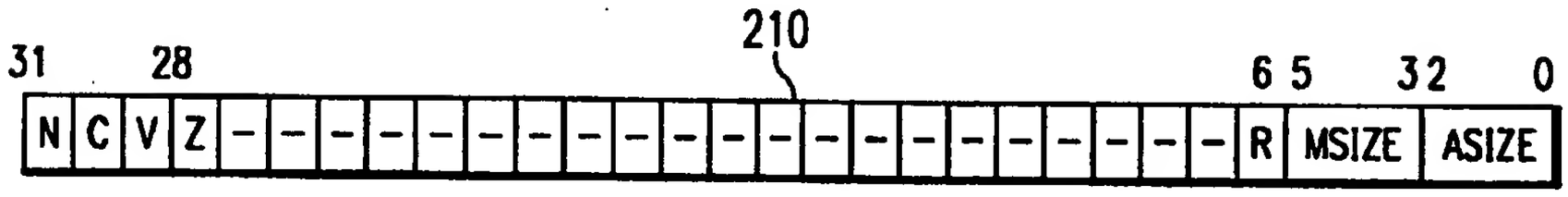


FIG. 6

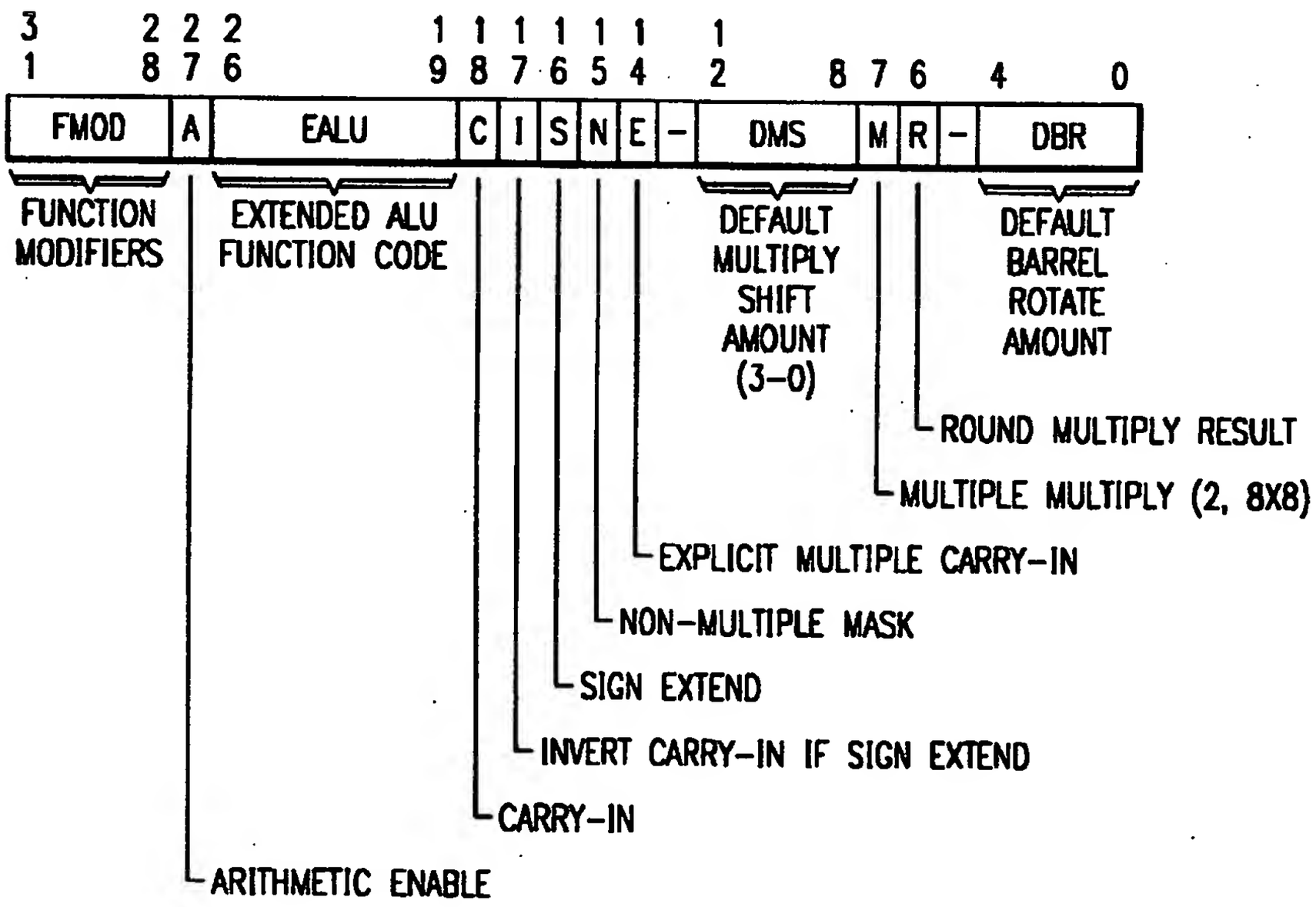


FIG. 9

	Document ID	U	Title	Current OR
1	US 5960465 A	☐	Apparatus and method for directly accessing compressed data utilizing a compressed memory address translation unit and compression descriptor table	711/208
2	US 5907374 A	☒	Method and apparatus for processing a compressed input bitstream representing an information signal	375/240.26
3	US 5610603 A	☒	Sort order preservation method used with a static compression dictionary having consecutively numbered children of a parent	341/51
4	US 5448301 A	☒	Programmable video transformation rendering method and apparatus	348/578
5	US 5361356 A	☒	Storage isolation with subspace-group facility	711/206
6	US 5351046 A	☒	Method and system for compacting binary coded decimal data	341/62
7	US 5088031 A	☒	Virtual machine file control system which translates block numbers into virtual addresses then into real addresses for accessing main storage	709/100
8	US 4389706 A	☒	Digital computer monitored and/or operated system or process which is structured for operation with an improved automatic programming process and system	700/1
9	US 4227245 A	☒	Digital computer monitored system or process which is configured with the aid of an improved automatic programming system	700/95
10	US 4216528 A	☒	Digital computer implementation of a logic director or sequencer	700/95
11	US 4215407 A	☒	Combined file and directory system for a process control digital computer system	700/95
12	US 4215406 A	☒	Digital computer monitored and/or operated system or process which is structured for operation with an improved automatic programming process	700/95

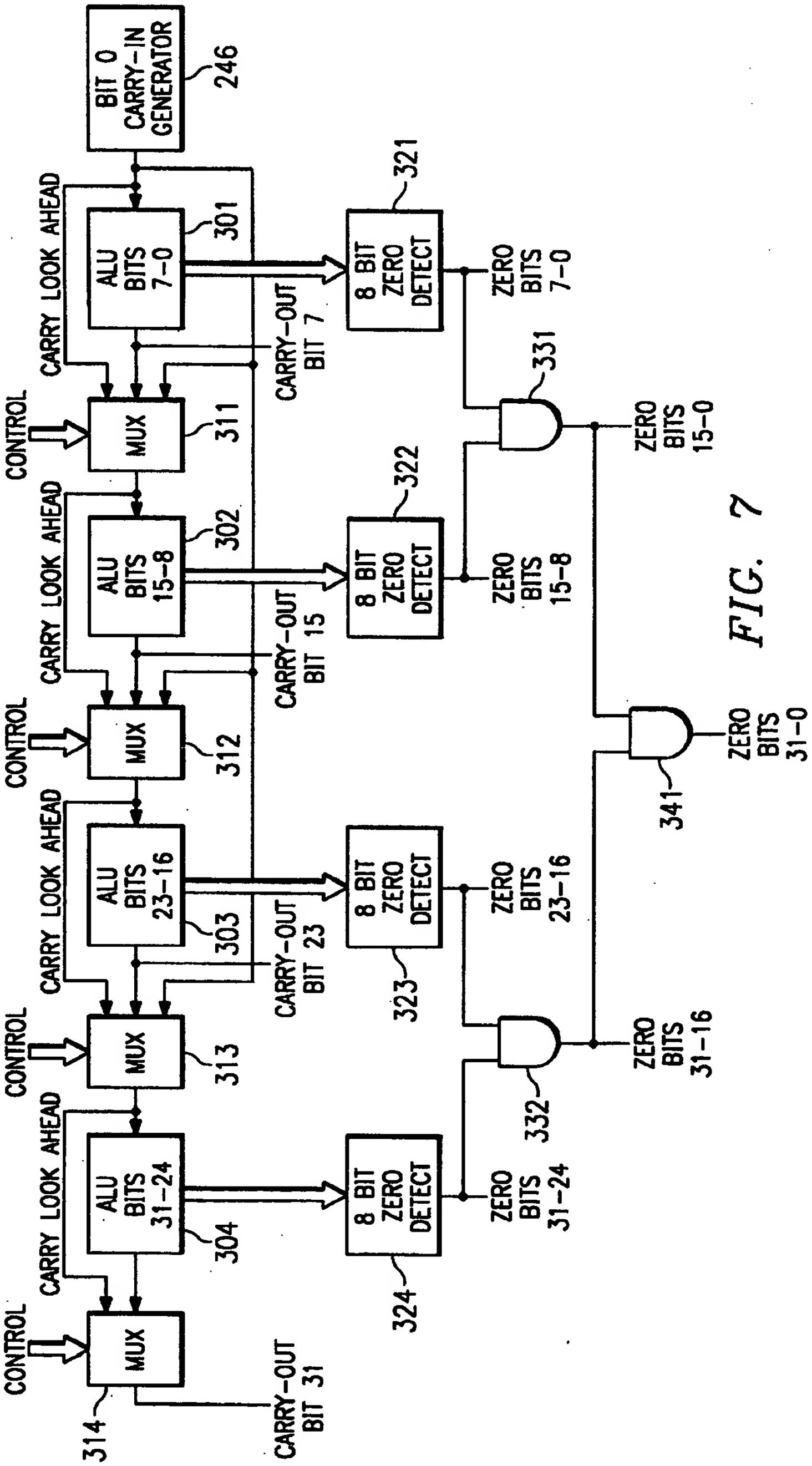

BIT 0 CARRY-IN GENERATOR
246
ALU BITS 7-0
301
8 BIT ZERO DETECT
321
ZERO BITS 7-0
CARRY LOOK AHEAD
CONTROL
MUX
311
CARRY-OUT BIT 7
ALU BITS 15-8
302
8 BIT ZERO DETECT
322
ZERO BITS 15-8
CARRY-OUT BIT 15
MUX
312
ALU BITS 23-16
303
8 BIT ZERO DETECT
323
ZERO BITS 23-16
CARRY-OUT BIT 23
MUX
313
ALU BITS 31-24
304
8 BIT ZERO DETECT
324
ZERO BITS 31-24
MUX
314
CARRY-OUT BIT 31
331
ZERO BITS 15-0
341
ZERO BITS 31-0
332
ZERO BITS 31-16

FIG. 7

	Document	U	Title	Current OR	S	C
1	USD 5809306 A	☐	Variable address length compiler and processor improved in address management	717/5	☒	☐

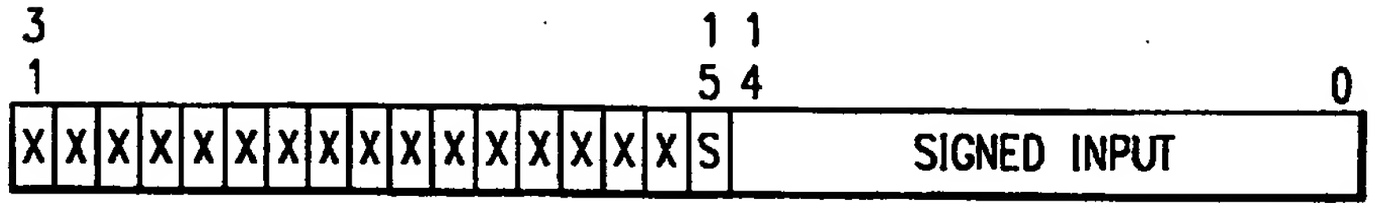


FIG. 10a

FIG. 10b ("01" IF HEX "8000" X HEX "8000")

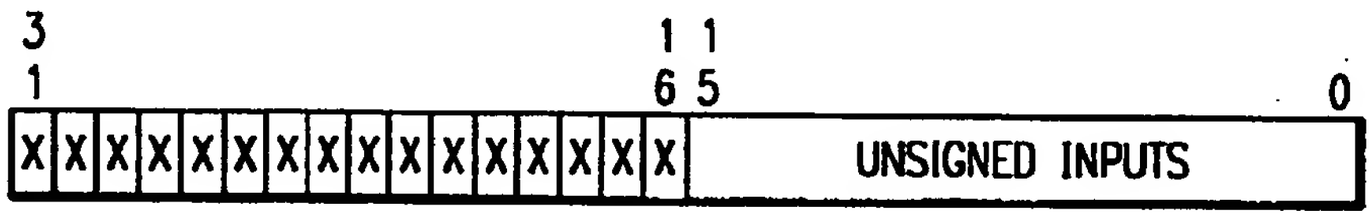


FIG. 10c

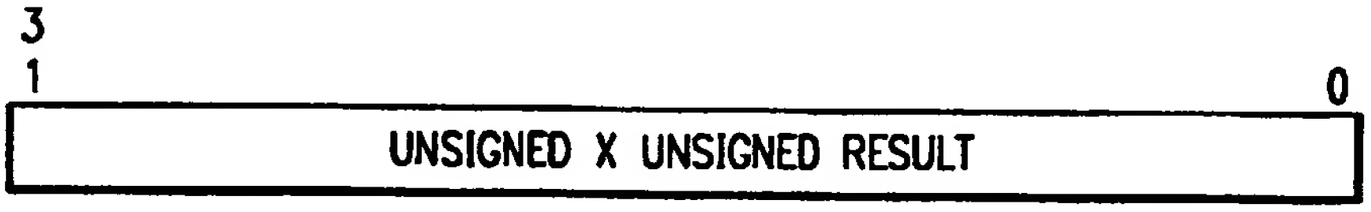


FIG. 10d

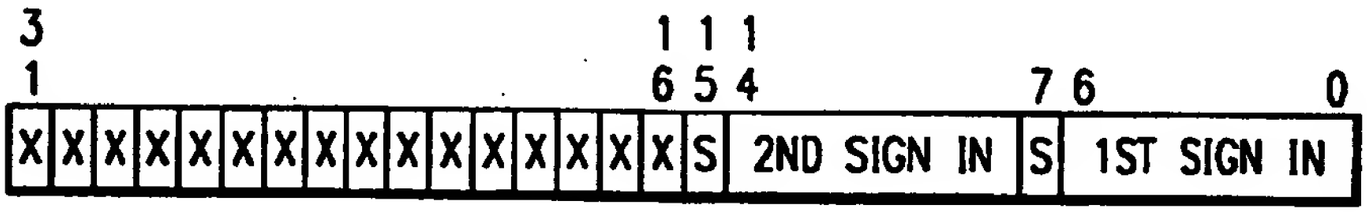


FIG. 11a

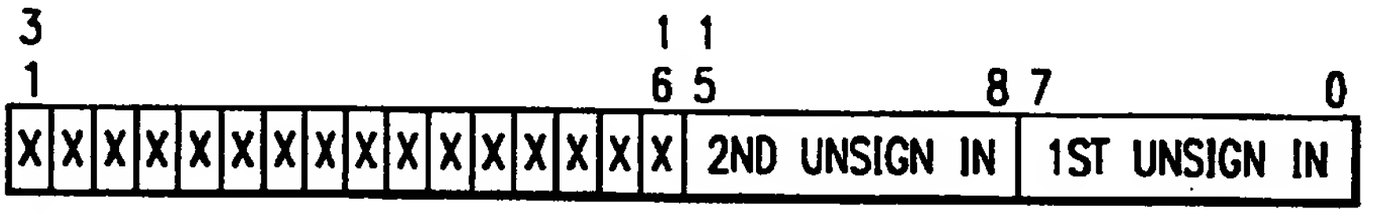


FIG. 11b

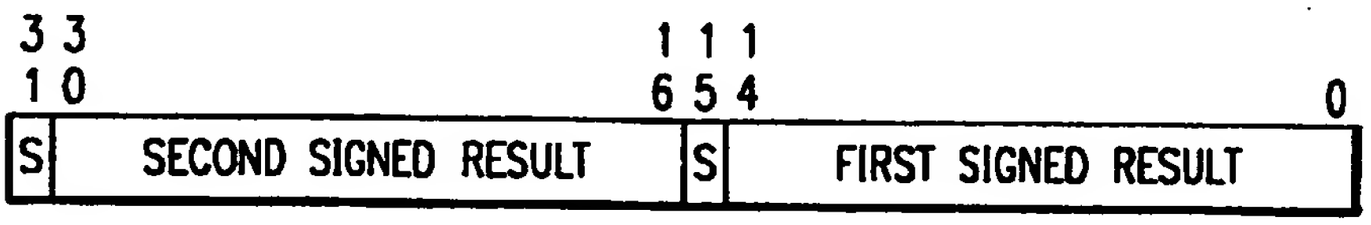


FIG. 11c

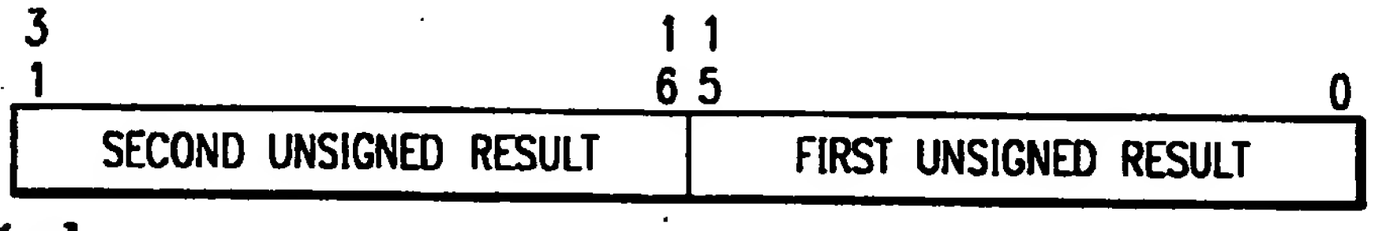


FIG. 11d

	Document	U	Title	Current OR
1	US3D 58093 06 A	☐	Variable address length compiler and processor improved in address management	717/5

FIG. 15a

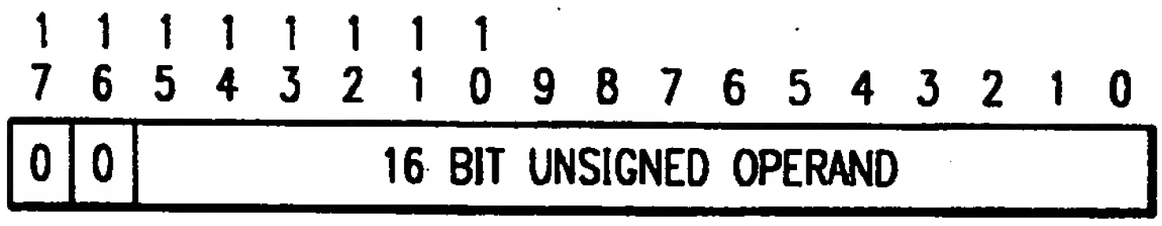


FIG. 15b

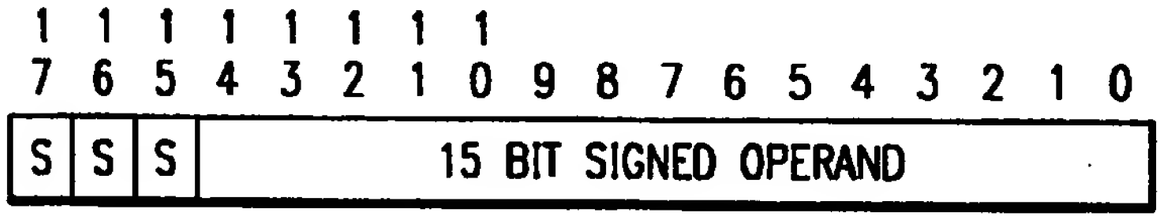


FIG. 16a

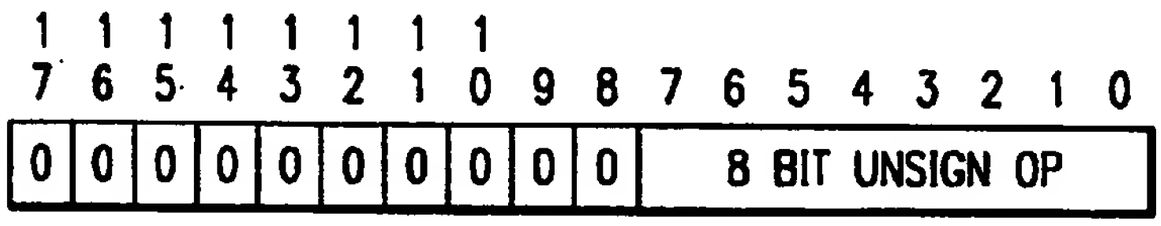


FIG. 16b

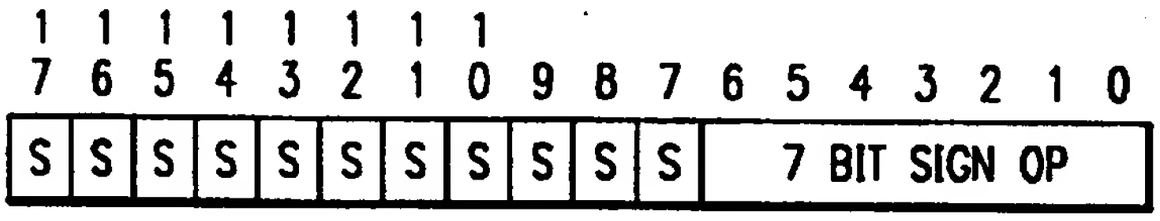


FIG. 16c

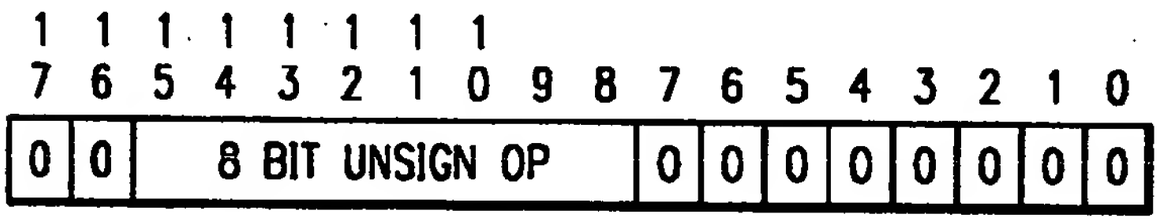


FIG. 16d

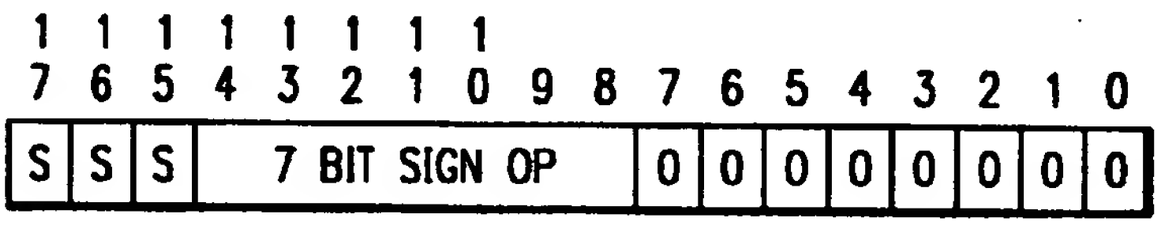


FIG. 17a

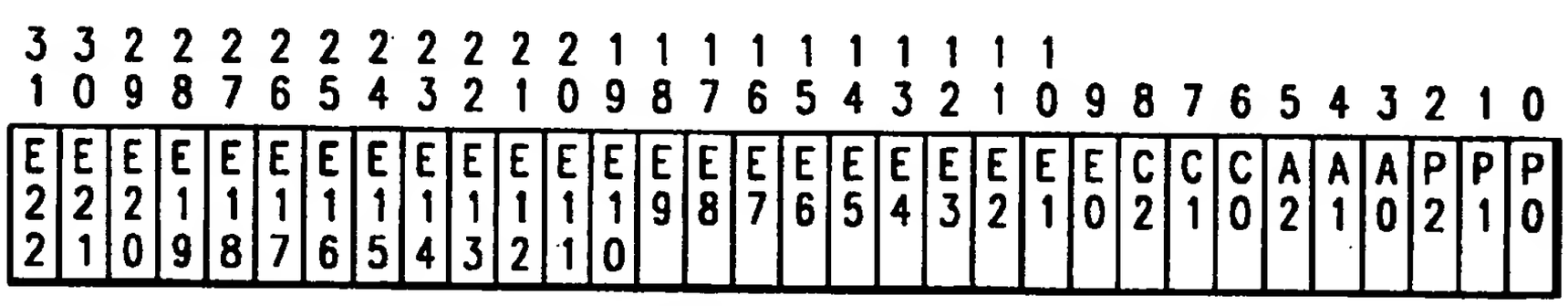
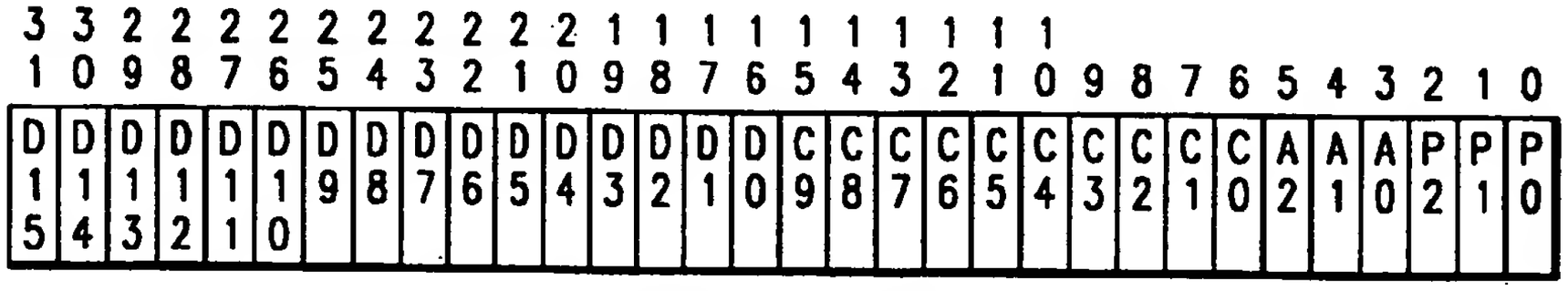


FIG. 17b



	Document	U	Title	Current OR
1	USD 61417 90 A	☐	Instructions signature and primary input and primary output extraction within an IEEE 1149.1 compliance checker	716/5
2	US 61417 42 A	☒	Method for reducing number of bits used in storage of instruction address pointer values	711/220
3	US 61311 54 A	☒	Microcomputer having variable bit width area for displacement	712/32
4	US 61227 24 A	☒	Central processing unit having instruction queue of 32-bit length fetching two instructions of 16-bit fixed length in one instruction fetch operation	712/41
5	US 61051 05 A	☒	Data processing system using configuration select logic, an instruction store, and sequencing logic during instruction execution	711/103
6	US 60932 13 A	☒	Flexible implementation of a system management mode (SMM) in a processor	703/27
7	US 60853 06 A	☒	Processor for executing highly efficient VLIW	712/24
8	US 60766 52 A	☒	Assembly line system and apparatus controlling transfer of a workpiece	198/341.07
9	US 60790 03 A	☒	Reverse TLB for providing branch target address in a microprocessor having a physically-tagged cache	711/200
10	US 60790 05 A	☒	Microprocessor including virtual address branch prediction and current page register to provide page portion of virtual and physical fetch address	711/213
11	US 60617 80 A	☒	Execution unit chaining for single cycle extract instruction having one serial shift left and one serial shift right execution units	712/204
12	US 60444 50 A	☒	Processor for VLIW instruction	712/24
13	US 60391 68 A	☒	Method of manufacturing a product from a workpiece	198/341.07
14	US 60410 10 A	☒	Graphics controller integrated circuit without memory interface pins and associated power dissipation	365/226
15	US 60353 90 A	☒	Method and apparatus for generating and logically combining less than (LT), greater than (GT), and equal to (EQ) condition code bits concurrently with the execution of an arithmetic or logical operation	712/220
16	US 60351 23 A	☒	Determining hardware complexity of software operations	717/9
17	US 60235 64 A	☒	Data processing system using a flash reconfigurable logic device as a dynamic execution unit for a sequence of instructions	703/23
18	US 60121 55 A	☒	Method and system for performing automatic extraction and compliance checking of an IEEE 1149.1 standard design within a netlist	714/727
19	US 60121 37 A	☒	Special purpose processor for digital audio/video decoding	712/36
20	US 59915 45 A	☒	Microcomputer having variable bit width area for displacement and circuit for handling immediate data larger than instruction word	712/33

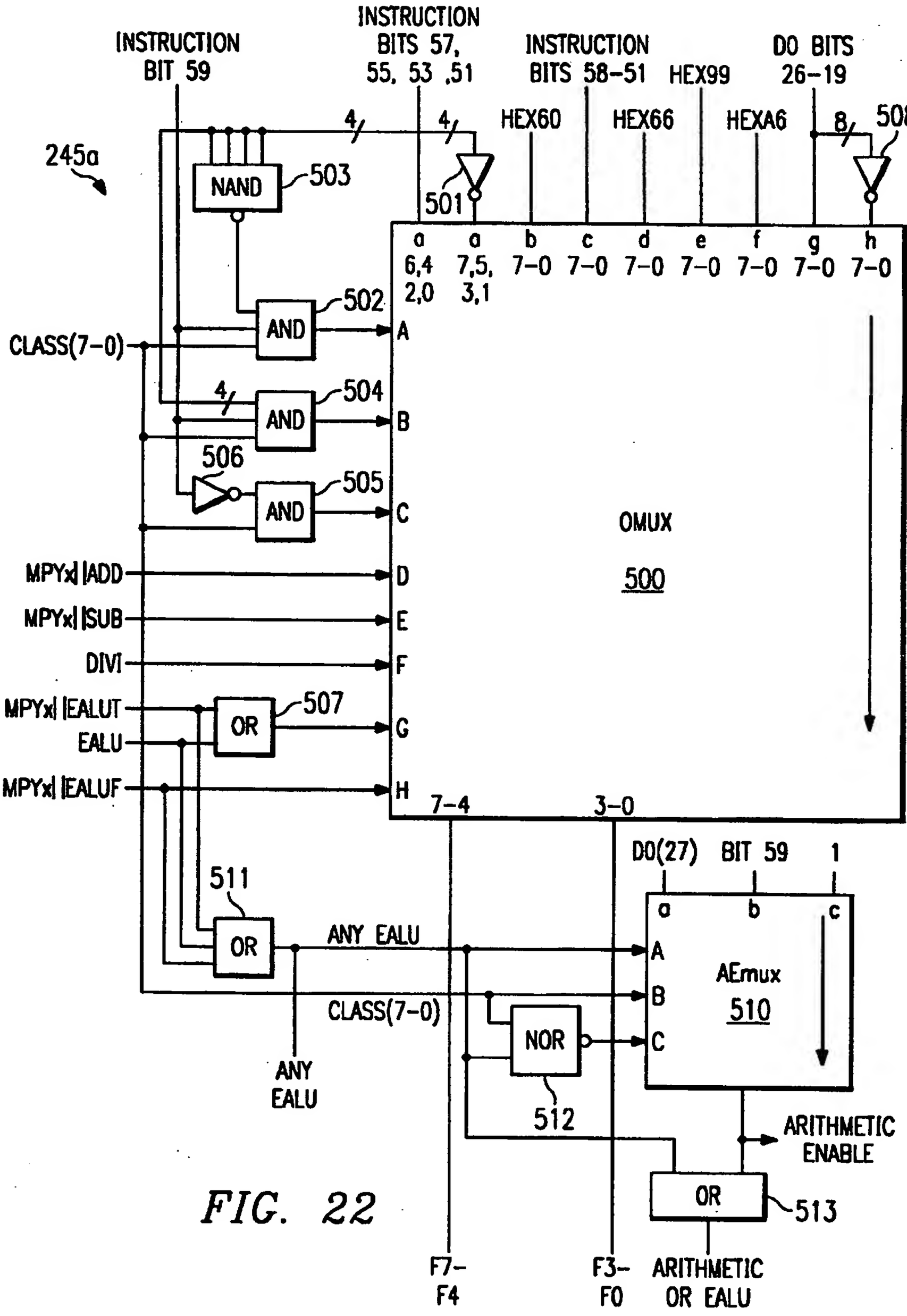

INSTRUCTION BIT 59
INSTRUCTION BITS 57, 55, 53 ,51
INSTRUCTION BITS 58-51
HEX99
DO BITS 26-19
HEX60
HEX66
HEXA6
508
245a
NAND
503
501
4/
4/
8/
a 6,4 2,0
a 7,5, 3,1
b 7-0
c 7-0
d 7-0
e 7-0
f 7-0
g 7-0
h 7-0
502
AND
A
CLASS(7-0)
504
AND
B
506
505
AND
C
OMUX
500
MPYx|ADD
D
MPYx|SUB
E
DIVI
F
MPYx|EALUT
EALU
OR
507
G
MPYx|EALUF
H
7-4
3-0
511
OR
ANY EALU
DO(27)
BIT 59
1
a
b
c
A
B
C
AEmux
510
CLASS(7-0)
NOR
ANY EALU
512
ARITHMETIC ENABLE
OR
513
F7-F4
F3-F0
ARITHMETIC OR EALU

FIG. 22

	Document	U	Title	Current OR
21	USD 5987488 A	☒	Matrix processor	708/520
22	US 5978937 A	☒	Microprocessor and debug system	714/45
23	US 5969976 A	☒	Division circuit and the division method thereof	708/655
24	US 5966514 A	☒	Microprocessor for supporting reduction of program codes in size	712/210
25	US 5938759 A	☒	Processor instruction control mechanism capable of decoding register instructions and immediate instructions with simple configuration	712/209
26	US 5930158 A	☒	Processor with instruction set for audio effects	708/204
27	US 5926642 A	☒	RISC86 instruction set	712/1
28	US 5920713 A	☒	Instruction decoder including two-way emulation code branching	712/236
29	US 5918045 A	☒	Data processor and data processing system	712/237
30	US 5909567 A	☒	Apparatus and method for native mode processing in a RISC-based CISC processor	712/208
31	US 5872965 A	☒	System and method for performing multiway branches using a visual instruction set	712/236
32	US 5854920 A	☒	Method and apparatus for manipulating a carry/borrow bit to numerically adjust and immediate value of an instruction during execution	712/221
33	US 5826072 A	☒	Pipelined digital signal processor and signal processing system employing same	712/226
34	US 5819058 A	☒	Instruction compression and decompression system and method for a processor	712/210
35	US 5819056 A	☒	Instruction buffer organization method and system	712/204
36	US 5809306 A	☒	Variable address length compiler and processor improved in address management	717/5
37	US 5809294 A	☒	Parallel processing unit which processes branch instructions without decreased performance when a branch is taken	712/233
38	US 5809273 A	☒	Instruction predecode and multiple instruction decode	712/210
39	US 5794063 A	☒	Instruction decoder including emulation using indirect specifiers	712/23
40	US 5771363 A	☒	Single-chip microcomputer having an expandable address area	712/200
41	US 5764990 A	☒	Compact encoding for storing integer multiplication Sequences	717/5

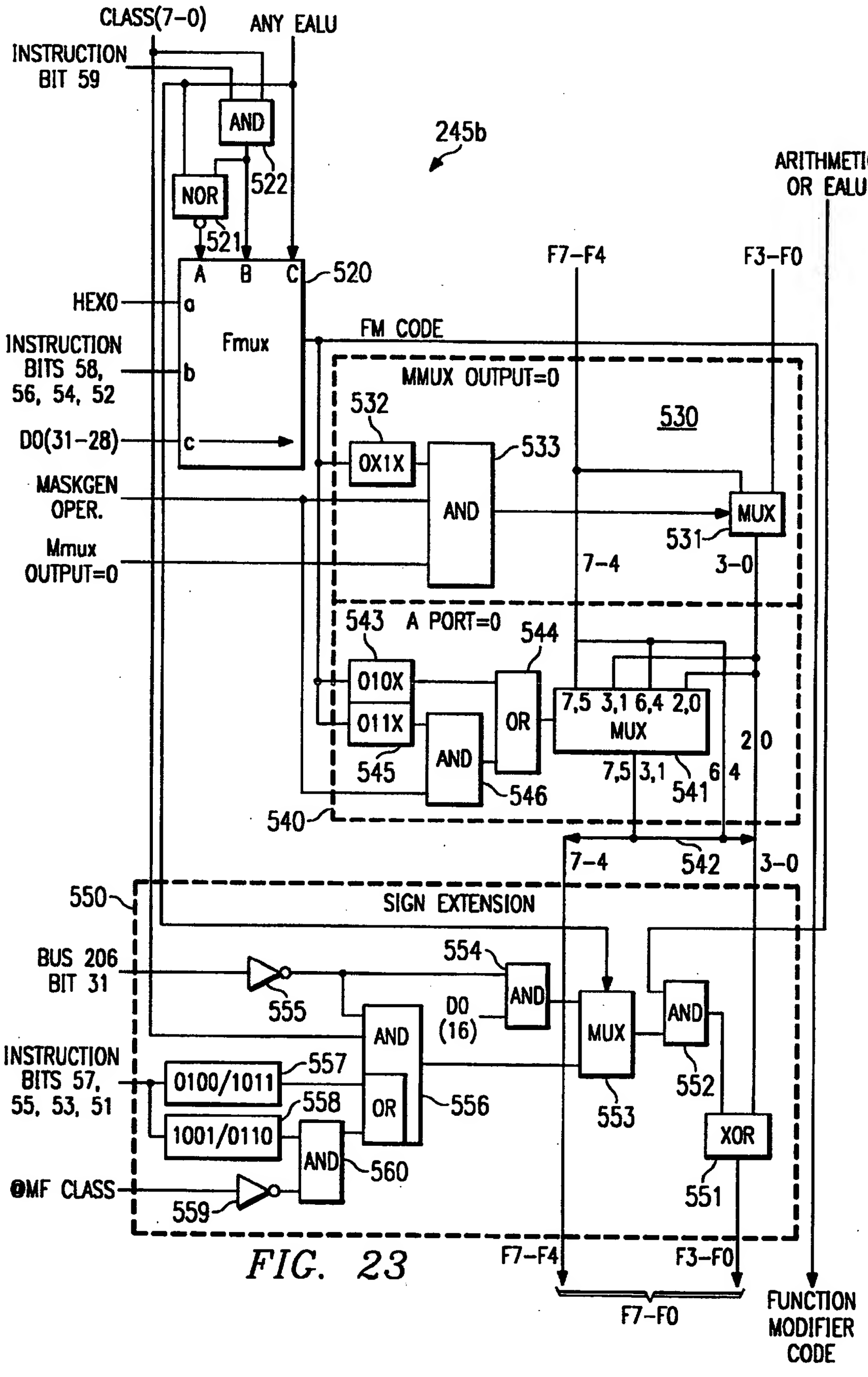

CLASS(7-0)
ANY EALU
INSTRUCTION BIT 59
AND
NOR
522
521
A B C
520
245b
ARITHMETIC OR EALU
F7-F4
F3-F0
HEX0
a
Fmux
INSTRUCTION BITS 58, 56, 54, 52
b
DO(31-28)
c
FM CODE
MMUX OUTPUT=0
532
530
0X1X
533
AND
MUX
531
MASKGEN OPER.
Mmux OUTPUT=0
7-4
3-0
543 A PORT=0
544
010X
011X
OR
7,5 3,1 6,4 2,0
MUX
545
AND
546
7,5 3,1
541
6 4
2 0
540
542
550
SIGN EXTENSION
BUS 206 BIT 31
554
555
AND
DO (16)
AND
MUX
AND
552
INSTRUCTION BITS 57, 55, 53, 51
557
0100/1011
AND
OR
556
558
1001/0110
AND
560
XOR
553
551
OMF CLASS
559
F7-F4
F3-F0
F7-F0
FUNCTION MODIFIER CODE

FIG. 23

	Document ID	U	Title	Current OR
42	US 5732255 A	☒	Signal processing system with ROM storing instructions encoded for reducing power consumpton during reads and method for encoding such instructions	712/248
43	US 5717616 A	☒	Computer hardware instruction and method for computing population counts	708/210
44	US 5687360 A	☒	Branch predictor using multiple prediction heuristics and a heuristic identifier in the branch instruction	712/240
45	US 5687344 A	☒	Single-chip microcomputer having an expandable address area	711/220
46	US 5682545 A	☒	Microcomputer having 16 bit fixed length instruction format	712/41
47	US 5615140 A	☒	Fixed-point arithmetic unit	708/518
48	US 5568646 A	☒	Multiple instruction set mapping	712/209
49	US 5568630 A	☒	Backward-compatible computer architecture with extended word size and address space	712/200
50	US 5491793 A	☒	Debug support in a processor chip	714/45
51	US 5473758 A	☒	System having input output pins shifting between programming mode and normal mode to program memory without dedicating input output pins for programming mode	711/103
52	US 5420992 A	☒	Backward-compatible computer architecture with extended word size and address space	703/27
53	US 5408620 A	☒	Circuit for executing conditional branch instructions in pipeline process	712/234
54	US 5381532 A	☒	Microprocessor having branch aligner between branch buffer and instruction decoder unit for enhancing initiation of data processing after execution of conditional branch instruction	712/237
55	US 5355508 A	☒	Parallel data processing system combining a SIMD unit with a MIMD unit and sharing a common bus, memory, and system controller	712/20
56	US 5355462 A	☒	Processor data memory address generator	711/215
57	US 5327536 A	☒	Microprocessor having branch prediction function	712/238
58	US 5301295 A	☒	Data processor apparatus and method with selective caching of instructions	711/125
59	US 5261076 A	☒	Method for programming a pin compatible memory device by maintaining a reset clock signal longer than a regular reset duration	712/200
60	US 5247627 A	☒	Digital signal processor with conditional branch decision unit and storage of conditional branch decision results	712/236
61	US 5237667 A	☒	Digital signal processor system having host processor for writing instructions into internal processor memory	712/248

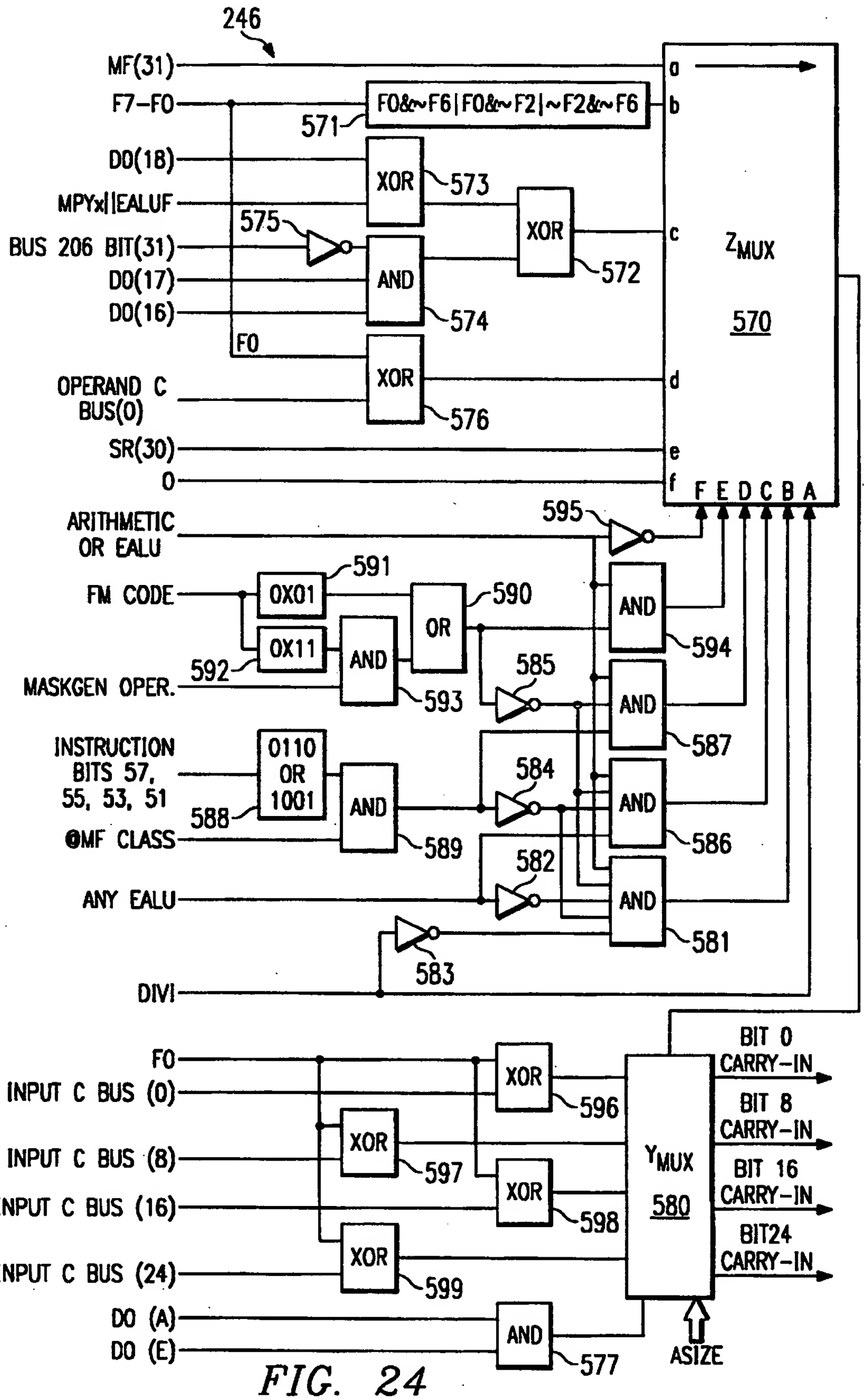

246
MF(31)
F7–F0
D0(18)
MPYx||EALUF
BUS 206 BIT(31)
D0(17)
D0(16)
F0
OPERAND C BUS(0)
SR(30)
0
ARITHMETIC OR EALU
FM CODE
MASKGEN OPER.
INSTRUCTION BITS 57, 55, 53, 51
MF CLASS
ANY EALU
DIVI
F0
INPUT C BUS (0)
INPUT C BUS (8)
INPUT C BUS (16)
INPUT C BUS (24)
D0 (A)
D0 (E)
F0&~F6|F0&~F2|~F2&~F6
571
XOR
573
XOR
572
575
AND
574
XOR
576
591
0X01
0X11
592
AND
OR
590
593
0110 OR 1001
588
AND
589
595
AND
594
585
AND
587
584
AND
586
582
AND
581
583
AND
577
596
XOR
597
XOR
598
XOR
599
a
b
c
d
e
f
F E D C B A
ZMUX
570
YMUX
580
BIT 0 CARRY–IN
BIT 8 CARRY–IN
BIT 16 CARRY–IN
BIT24 CARRY–IN
ASIZE

FIG. 24

	Document	U	Title	Current OR
62	USD 52222 41 A	☒	Digital signal processor having duplex working registers for switching to standby state during interrupt processing	712/228
63	US 52166 13 A	☒	Segmented asynchronous operation of an automated assembly line	700/102
64	US 52108 33 A	☒	System for selectively masking data in a branch address register and replacing the microinstruction address register by the masked data	712/236
65	US 52069 40 A	☒	Address control and generating system for digital signal-processor	711/218
66	US 51704 75 A	☒	Data processor having instructions for interpolating between memory-resident data values respectively	712/223
67	US 51347 11 A	☒	Computer with intelligent memory system	712/17
68	US 50705 00 A	☒	Memory package system utilizing inductive coupling between memory module and read/write unit	365/189.02
69	US 50459 93 A	☒	Digital signal processor	712/236
70	US 49841 54 A	☒	Instruction prefetching device with prediction of a branch destination address	712/240
71	US 49758 28 A	☒	Multi-channel data communications controller	710/11
72	US 49473 69 A	☒	Microword generation mechanism utilizing a separate branch decision programmable logic array	712/234
73	US 49071 92 A	☒	Microprogram control unit having multiway branch	712/236
74	US 48902 23 A	☒	Paged memory management unit which evaluates access permissions when creating translator	711/207
75	US 48846 74 A	☒	Segmented asynchronous operation of an automated assembly line	198/341.01
76	US 48826 73 A	☒	Method and apparatus for testing an integrated circuit including a microprocessor and an instruction cache	714/42
77	US 48004 89 A	☒	Paged memory management unit capable of selectively supporting multiple address spaces	711/206
78	US 47854 52 A	☒	Error detection using variable field parity checking	714/756
79	US 47665 37 A	☒	Paged memory management unit having stack change control register	711/163
80	US 47632 50 A	☒	Paged memory management unit having variable number of translation table levels	711/208
81	US 47632 44 A	☒	Paged memory management unit capable of selectively supporting multiple address spaces	711/206
82	US 46023 47 A	☒	Microcomputer addressing system and electronic timepiece utilizing the same	702/178
83	US 45063 25 A	☒	Reflexive utilization of descriptors to reconstitute computer instructions which are Huffman-like encoded	341/65

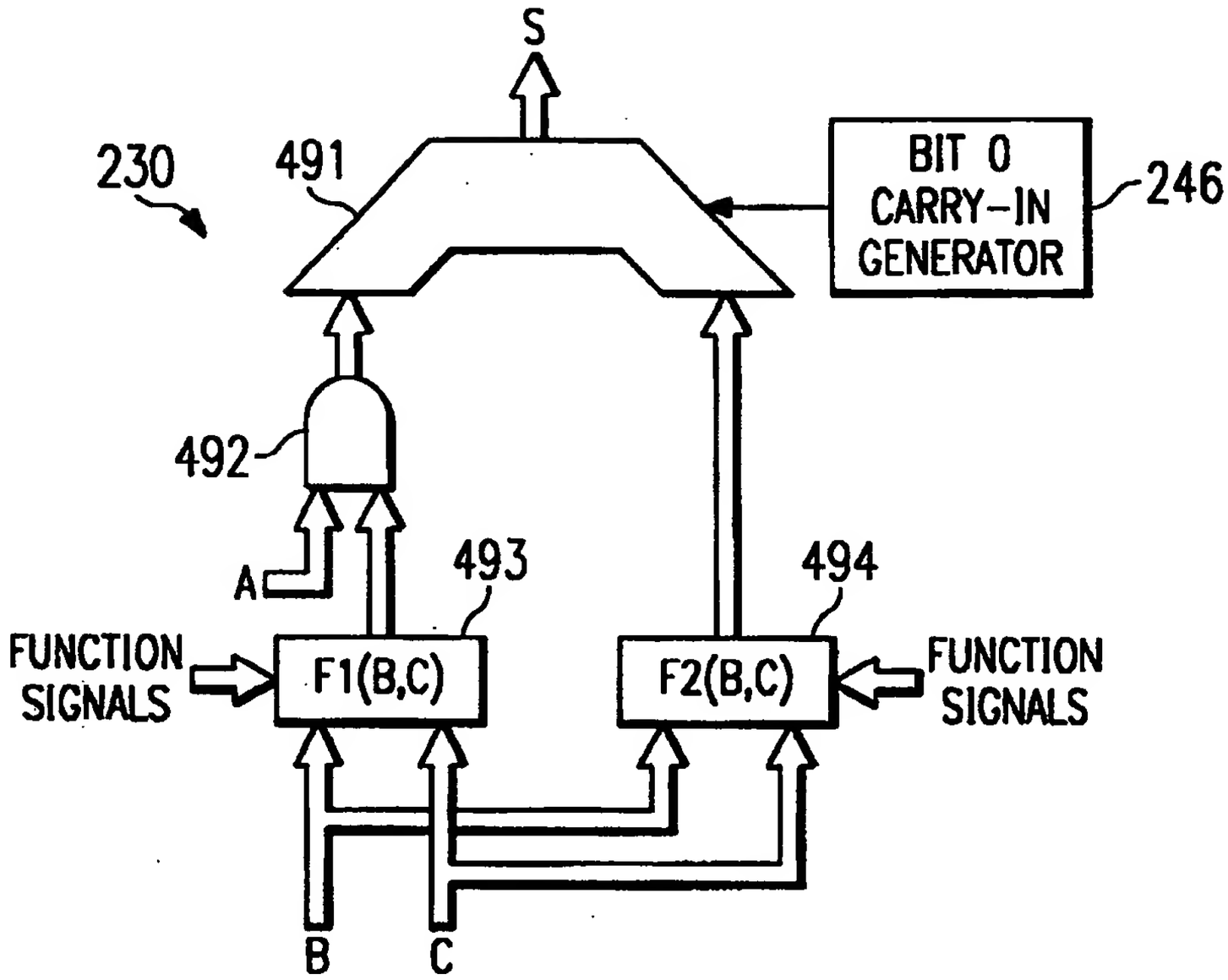

S
230
491
BIT 0
CARRY-IN
GENERATOR
246
492
A
493
494
FUNCTION
SIGNALS
F1(B,C)
F2(B,C)
FUNCTION
SIGNALS
B
C

FIG. 25

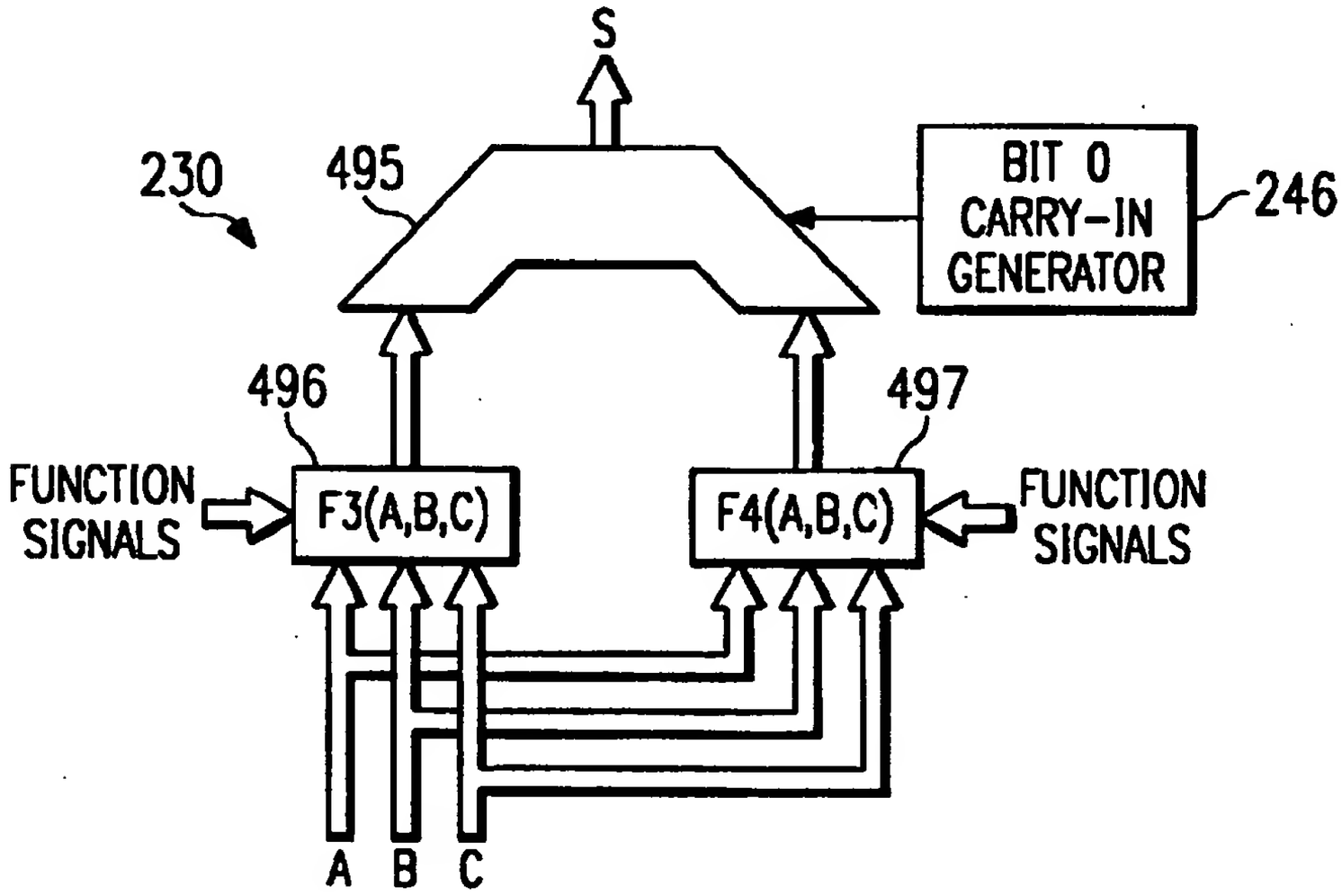

S
230
495
BIT 0
CARRY-IN
GENERATOR
246
496
497
FUNCTION
SIGNALS
F3(A,B,C)
F4(A,B,C)
FUNCTION
SIGNALS
A
B
C

FIG. 26

	Document	U	Title	Current OR
84	USD 45049 04 A	☒	Binary logic structure employing programmable logic arrays and useful in microword generation apparatus	326/41
85	US 43934 69 A	☒	Process control apparatus	712/234
86	US 43618 68 A	☒	Device for increasing the length of a logic computer address	711/2
87	US 43576 38 A	☒	Index information recording method for picture information filing system	360/72.2
88	US 43143 42 A	☒	Unsafe machines without safe positions	700/100
89	US 43062 92 A	☒	Segmented asynchronous operation of an automated assembly line	700/100
90	US 43062 85 A	☒	Data processing apparatus	712/226
91	US 42662 53 A	☒	Processor for a graphic terminal	358/425
92	US 42492 48 A	☒	Programmable sequence controller with an arithmetic operation function	345/430
93	US 38630 59 A	☒	FAULT RECOGNITION APPARATUS FOR ARITHMETIC/LOGIC DEVICES	708/530
94	US 37741 66 A	☒	SHORT-RANGE DATA PROCESSING TRANSFERS	711/200
95	US 37532 34 A	☒	MULTICOMPUTER SYSTEM WITH SIMULTANEOUS DATA INTERCHANGE BETWEEN COMPUTERS	709/253
96	US 37131 08 A	☒	BRANCH CONTROL FOR A DIGITAL MACHINE	712/234
97	US 36263 74 A	☒	HIGH-SPEED DATA-DIRECTED INFORMATION PROCESSING SYSTEM CHARACTERIZED BY A PLURAL-MODULE BYTE-ORGANIZED MEMORY UNIT	711/5

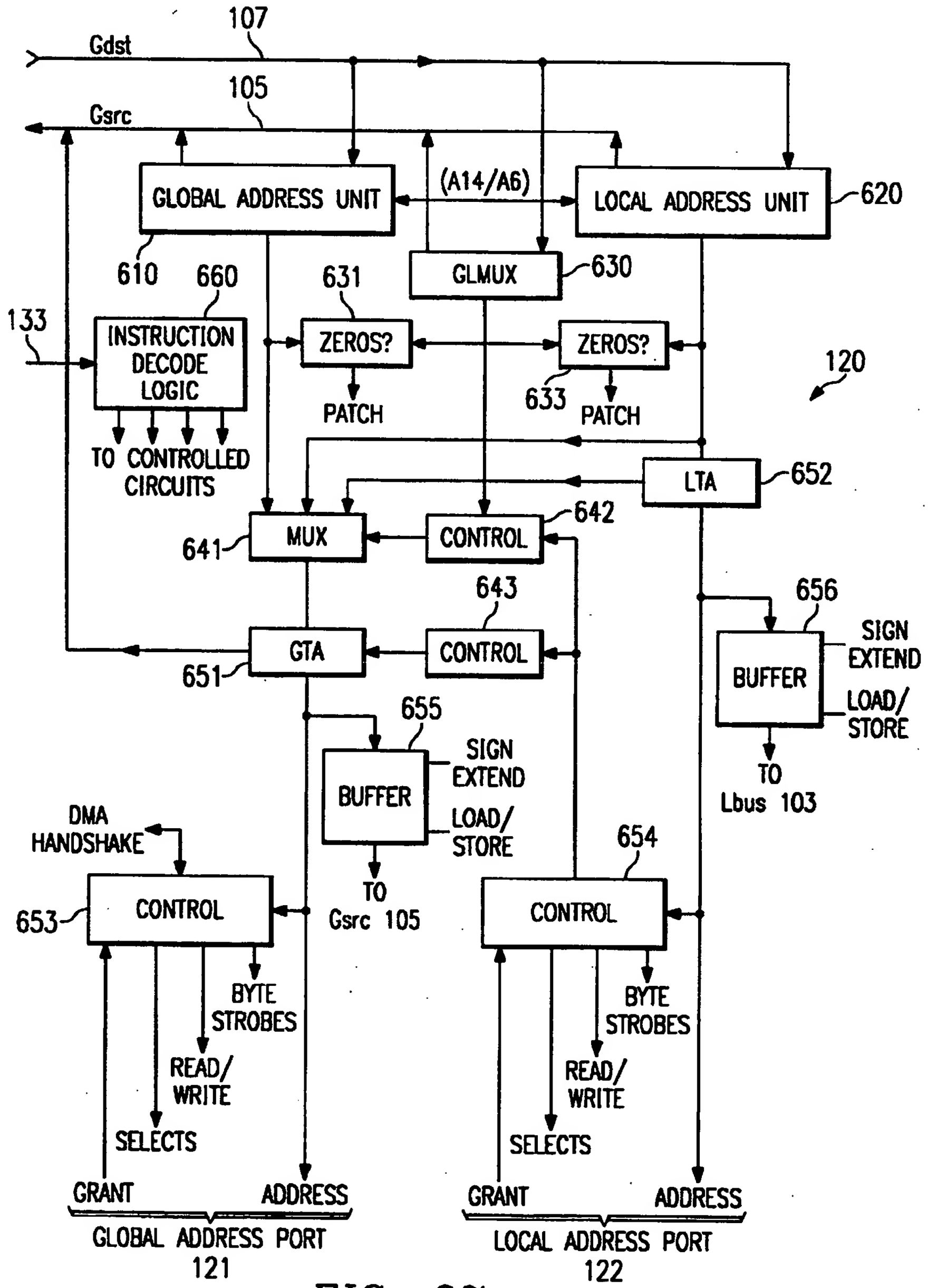

107
Gdst
105
Gsrc
GLOBAL ADDRESS UNIT
(A14/A6)
LOCAL ADDRESS UNIT
620
610
660
631
GLMUX
630
133
INSTRUCTION DECODE LOGIC
ZEROS?
ZEROS?
120
PATCH
633
PATCH
TO CONTROLLED CIRCUITS
LTA
652
MUX
641
CONTROL
642
643
CONTROL
GTA
651
656
BUFFER
SIGN EXTEND
LOAD/ STORE
TO Lbus 103
655
BUFFER
SIGN EXTEND
LOAD/ STORE
DMA HANDSHAKE
654
TO Gsrc 105
CONTROL
653
CONTROL
BYTE STROBES
READ/ WRITE
SELECTS
BYTE STROBES
READ/ WRITE
SELECTS
GRANT
ADDRESS
GRANT
ADDRESS
GLOBAL ADDRESS PORT 121
LOCAL ADDRESS PORT 122

FIG. 27

	Document	U	Title	Current OR
1	US 5544247 A	☐	Transmission and reception of a first and a second main signal component	381/27
2	US 5481643 A	☒	Transmitter, receiver and record carrier for transmitting/receiving at least a first and a second signal component	704/227
3	US 5463746 A	☒	Data processing system having prediction by using an embedded guess bit of remapped and compressed opcodes	712/240
4	US 4835679 A	☒	Microprogram control system	712/212

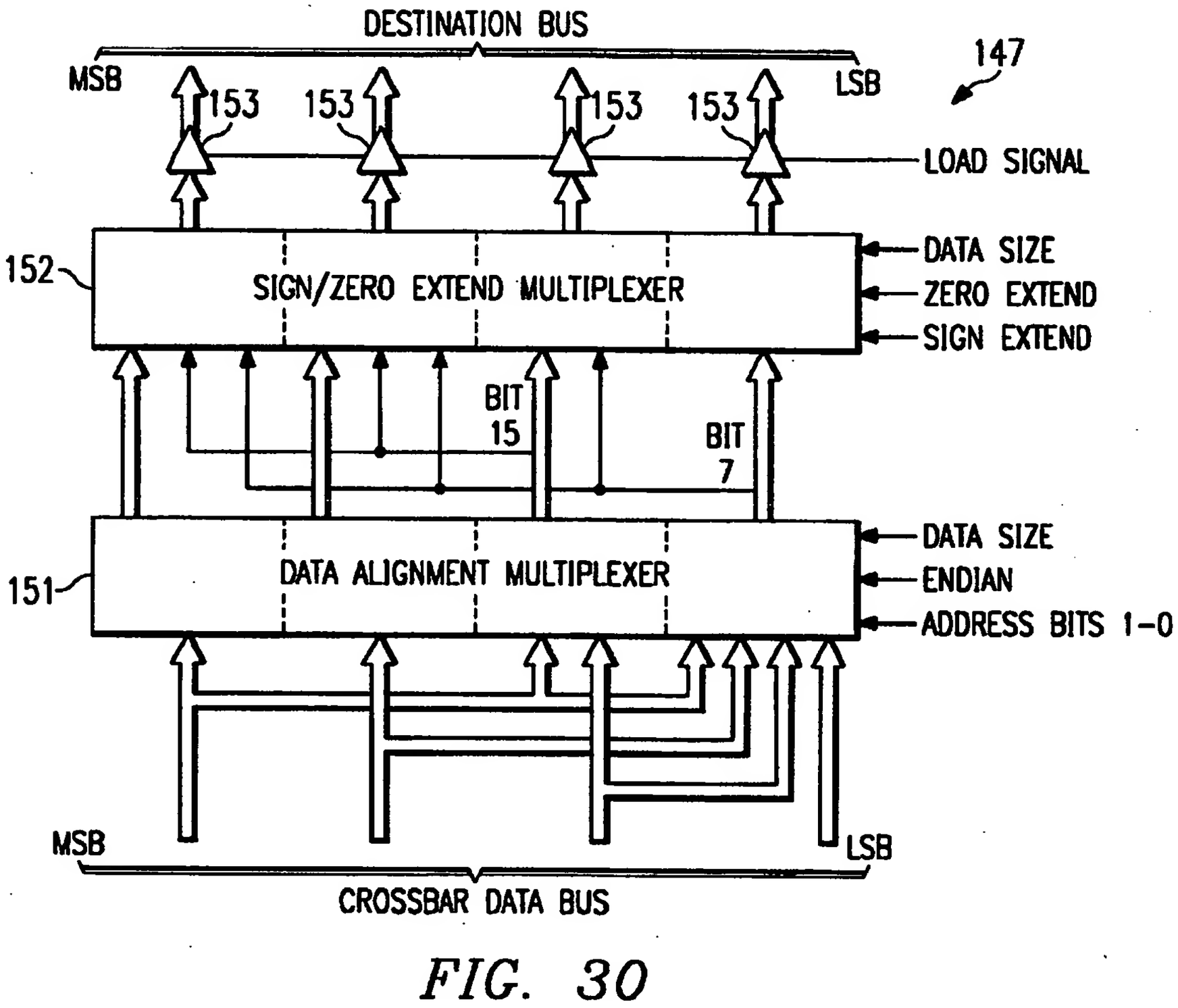

DESTINATION BUS
MSB
LSB
147
153
153
153
153
LOAD SIGNAL
152
SIGN/ZERO EXTEND MULTIPLEXER
DATA SIZE
ZERO EXTEND
SIGN EXTEND
BIT 15
BIT 7
151
DATA ALIGNMENT MULTIPLEXER
DATA SIZE
ENDIAN
ADDRESS BITS 1-0
MSB
LSB
CROSSBAR DATA BUS

FIG. 30

	L #	Hits	Search Text
1	L1	187	abbreviat$3 near5 instruction
2	L3	781	compress$3 near5 instruction
3	L4	286	compact$3 near5 instruction
4	L5	83	(expand$3 decopmress$3 conver$4) near20 3
5	L6	23	(expand$3 conver$4) near20 4
6	L7	7	(expand$3 conver$4) near20 1

Instruction Decode Stage

The instruction decode stage (D stage) **22** decodes an instruction code inputted from the IF stage **21**. Decoding is performed once per clock cycle by using an instruction decoder **75** such as a FHW (First Half Word) decoder, a NFHW (Not First Half Word) decoder, an addressing mode decoder or the like of the instruction decode unit **40**, and an instruction code of 0–six bytes is consumed in one-time decode processing (no instruction code is consumed in output processing or the like of the step code containing the return address of the return subroutine instruction). An A code, containing address calculation information and a D code, containing being an intermediate result of decode of the operation code are outputted to the A stage **23** by a one-time decode.

In the D stage **22**, control of the PC calculation unit **42** for each instruction and output processing of an instruction code from the instruction queues **72** and **73** are also performed.

In the D stage **22**, preceding jump instruction processing (D stage preceding jump) is performed for a jump instruction. The D code and the A code are not outputted in the case of a jump instruction having performed a preceding jump, except for a conditional branch instruction. Processing of the instruction is completed in the D stage **22**.

When a conditional branch instruction has been decoded, in the D stage **22**, the IF stage **21** is directed to fetch instructions from both of the branch destination and the non-branch destination. The instruction to be decoded following the conditional branch instruction is determined according to the result of branch prediction. This means that when the conditional branch instruction is predicted to cause a branch, the instruction outputted from the instruction queue A **72**, fetching the instruction of the branch destination is decoded, when the conditional instruction is predicted not to branch, the instruction code outputted from the instruction queue B **73** fetching the non-branch destination instruction is decoded.

Operand Address Calculation Stage

The operand address calculation stage (A stage) **23**, is divided roughly into two processings. One is a processing performing latter-stage decode of an operation code by using the decoder **76** of the instruction decode unit **40**, and the other is a processing performing calculation of an operand address in the operand address calculation unit **41**.

The latter-stage decode processing of the operation code inputs the D code, and performs write reservation to a register or a memory and output of a R code containing an entry address of a microprogram routine, parameters for the microprogram and the like. The write reservation to a register or a memory is to prevent the content of the register or the memory referred in address calculation from being rewritten by the preceding instruction on the pipeline such that a wrong calculation is performed.

The operand address calculation processing inputs the A code, performs address calculation of an operand in the operand address calculation unit **41** according to the A code, and outputs the result of the calculation as an F code. Also, for a jump instruction, it performs calculation of the jump destination address. It makes a check for write reservation in reading out the register being attended with address calculation, and when it determines that a reservation is present because the preceding instruction has not completed write processing to the register or the memory, it waits until the preceding instruction completes the write processing in the E stage **27**.

In the A stage **23**, a preceding jump processing (A stage preceding jump) is performed for a jump instruction which has not performed a preceding jump in the D stage **22**. For a register indirect jump and a memory indirect jump, the A stage preceding jump is performed. For an instruction which has performed an A stage preceding jump, the R code and the F code are not outputted, and the processing of the instruction is completed in the A stage **23**.

Micro-ROM Access Stage

The operand fetch stage (F stage) **26** is also divided roughly into two processings. One is an access processing of the micro-ROM, particularly being called the R stage **24**. The other is an operand prefetch processing, particularly being called the OF stage **25**. The R stage **24** and OF stage **25** are not always operated simultaneously, and their operational timings differ depending on miss or hit of the data cache, miss or hit of the data TLB or the like.

The micro-ROM access processing which occurs in the R stage **24** is a micro-ROM access and a microinstruction decode processing for producing an E code which is an execute control code to be used for execution in the next E stage **27** for the code $.

Where processing for one R code is decomposed into two or more microprogram steps, the IROM unit **43** and the FROM unit **44** are used in the E stage **27**, and the next R code is sometimes put in a micro-ROM-access-wait state. A micro-ROM access for the R code is performed when no micro-ROM access in the E stage **27** is performed. In the microprocessor, a number of integer operation instructions are performed in one microprogram step, and a number of floating point operation instructions are performed in two microprogram steps. Therefore, micro-ROM accesses for the R coded are often performed one after another.

In the processing of the R stage **24**, only the IROM unit **43** is accessed for an instruction not using the floating point processing unit **46**, and the FROM unit **44** is not accessed. The IROM unit **43** and the FROM unit **44** are both accessed for an instruction using the floating point processing unit **46** (floating point operation instruction, integer multiply/divide instruction or the like).

Operand Fetch Stage

The operand fetch stage (OF stage) **25** performs operand pre-fetch processing among the above-mentioned two processings to be performed in the F stage **26**.

In the operand fetch stage **25**, the logical address of the F code is translated into a physical address by the data TLB, the built-in data cache is accessed by the physical address, an operand is fetched, and the operand and the logical address transferred as an F code are combined and outputted as a S code.

In one F code an 8-byte boundary may be crossed, but an operand fetch of eight bytes or less is specified. The F code specifies of whether or not access of the operand is to be performed. Where the operand address itself and the immediate value which have been calculated in the A stage **23** are transferred to the E stage **27**, no operand pre-fetch is performed, and the content of the F code is transferred as an S code.

Execution Stage

The execution stage (E stage) **27** is operated with the E code and the S code taken as input. The E stage **27** executes an instruction, and the processings performed in the stages before the F stage **26** are all processings for the E stage **27**. When a jump is executed or a EIT processing is started in the E stage **27**, processings in the IF stage **21** through the E stage **27** are all invalid. The E stage **27** is controlled by a microprogram, and an instruction is executed by executing a series of micro-instructions from the entry address of a microprogram routine contained in the R code.

	Document	U	Title	Current OR
1	US 5938759 A	☐	Processor instruction control mechanism capable of decoding register instructions and immediate instructions with simple configuration	712/209
2	US 5897633 A	☒	System for converting programs and databases to correct year 2000 processing errors	707/6
3	US 5809500 A	☒	System for converting programs and databases to correct year 2000 processing errors	707/6
4	US 5634084 A	☒	Abbreviation and acronym/initialism expansion procedures for a text to speech reader	704/260
5	US 5548573 A	☒	Optical information reproducing apparatus provided with laser power control means for detecting reflected light from data region	369/116
6	US 4844469 A	☒	Golf trainer for calculating ball carry	473/225
7	US 4780810 A	☒	Data processor with associative memory storing vector elements for vector conversion	712/6

The logical address sent to the address translater **54** in this manner is first referred to the TLB **55**, and where the TLB has hit and a physical address is generated, the physical address is outputted to the data cache **56**. Where the TLB **55** has missed, in the address translater **54** an external bus access request for referring to an address translation table on the external memory is outputted to the bus interface unit **50**. Then, the reference to the address translation table is completed, and the generated physical addresses is outputted to the data cache **56**.

When the data cache **56** has hit, the data read out from the data cache **56** is registered into a S code register **60**, acting as an operand pre-fetch queue. The E stage **27** reads an operand from the S code register **60**, and performs processing. When a cache miss has taken place, a register request for the block having missed is sent to the bus interface unit **50**, and the data stored from the outside is inputted to the data cache **56**, being inputting also to the S code register **60**.

The following describes a processing sequence in the case where a store request is sent from the E stage **27**. The E stage **27** writes store data to a DD register **58**, writes a store address to the AA register **59**, and outputs an access request. When the access request is received by the OAU **48**, the address translater **54**, the TLB **55** and the data cache **56** are operated like the processing sequence after receiving an access from the above mentioned address calculation stage **29**. However, in the case of store processing, when the data cache **56** has hit, then store data must be written to the data cache **56**. When the data cache **56** has missed, nothing is performed.

The data cache **56** of the microprocessor uses a write-through system, and the data is required to be written also to the external memory. In general, the external bus cycle is slower than the machine cycle of the microprocessor, and therefore a storing buffer **57** is installed to prevent the internal processing from being speed-regulated by external bus access. The storing buffer **57** has three entries, and even when store data sent from the E stage **27** strides over a word boundary, the data can be registered into the storing buffer **57** in one time.

FIG. **8** is a block diagram showing a configuration of the data cache **56** of the present invention. This data cache **56** is operated in a four-way set associative system of 0th-3rd way, and the number of entries of each way is 64. The data cache **56** is composed of a first address register **101** temporarily storing an address given through the address bus, a second address register **102** temporarily storing part (six bits of lower order) of the address, a transferring path **112** connecting the first address register **101** and the second address register **102**, a tag entry decoder **103** which decodes six bits of lower order of the address and selects an entry of a tag memory **104**, the tag memory **104**, a data entry decoder **105** selecting an entry of a data memory **106**, the data memory **106**, a comparator **107** and a gate **108**.

Where reading data from an external memory **110** is performed as a result of processing an instruction, the address of the data to be read is sent to the data cache **56** through an address bus **100**, and the total bit width of the address (32 bits) is stored in the first address register **101** for the tag memory **104**. The six lower order address bits are stored in the second address register **102** for data. Then, the six lower order address bits of the first and the second address register **101** and **102** are sent respectively to the tag entry decoder **103** for the tag memory **104** and to the data entry decoder **105** for the data memory **106**, and a signal for selecting any of 64 entries is generated.

Next, in the tag memory **104**, four tags are read out from the selected entry, and the comparator **107** compares the four tags with higher-order bits excluding the above-mentioned six lower order bits of the first address register **101**. The corresponding data read out from the data memory **106** is selected by the gate **108**, and the data is outputted to the processing unit. Where data is read out from the data cache **56** in such a manner, the tag- and entry-decoders **103** and **105** perform identical operations.

The following provides a description of writing data to the external memory **110** as a result of processing an instruction. In the case of write processing, an address sent through the address bus **100** is stored only into the first address register **101**. Then, like read processing, retrieval of the tag memory **104** is performed. When the data read out from the tag memory **104** and the higher-order bits of the first address register **101** coincide with each other, six bits of lower order of the first address register **101** are transferred to the second address register **102**. If an address for write processing is sent in the next request, the address is stored into the fist address register **101**.

Next, in the data memory **106**, the transferred lower-order address is decoded by the data entry decoder **105**, and a signal for selecting any of 64 entries is generated. Then, based on the way information of the tag memory **104**, data is written to the corresponding data memory **106** from an internal data bus **111**. On the other hand, the tag memory **104** performs a retrieval of an address corresponding to the next write processing, and when they coincide with each other, six lower order bits of the first address register **101** are transferred to the second address register **102** like the above case.

FIG. **9** shows a timing chart of the microprocessor having the built-in data cache **56** of the present invention. FIG. **9** shows the case where a first operation is a read, and the second to fourth operations are writes. First, the first operation is a read, and therefore the address thereof is written simultaneously to the first address register **101** for the tag memory **104** and the second address register **102** for the data memory **106**.

Then, a write processing comes next, and therefore the address is written only to the first address register **101** for the tag memory **104**, and is not written to the second address register **102** for the data memory **106**. Then, when the address of write has coincided with the tag information (cache hit), the address is transferred to the second address register **102**. At this time, when it is a cache miss, write to the data memory **106** of the data cache **56** is not required, and therefore the address of the next read processing can also be stored into the first address register **101** and the second address register **102**. The above processing enables the cache to make a cache access overlap in one clock period for consecutive writes in a cache hit or miss and for a read following a write at a cache miss.

Also, the entry decoders **103** and **105** are provided independently for the tag memory **104** and the data memory **106**, and therefore a bus snooping operation can also be performed in parallel with a cache write. In a case where the microprocessor is connected to another microprocessor on an external bus, and accesses a common external memory (main memory), coherency of the respective data caches **56** is required to be held, and in the case where the other microprocessor renews the main memory, the block of the data cache **56** having the area must be invalid. For this reason, the first address register **101** for the tag memory **104** must have a function of inputting an address from the inside, and monitoring and storing an external address. In this microprocessor, in such a case, even if the first address register **101** and the tag entry decoder **103** are used for bus

	Document	U	Title	Current OR
1	US 6101592 A	☐	Methods and apparatus for scalable instruction set architecture with dynamic compact instructions	712/20
2	US 6027428 A	☒	Automated method and apparatus for providing real time personal physical fitness instruction	482/4
3	US 6006179 A	☒	Audio codec using adaptive sparse vector quantization with subband vector classification	704/222
4	US 5935256 A	☒	Parallel processing integrated circuit tester	713/400
5	US 5931953 A	☒	Parallel processing integrated circuit tester	713/500
6	US 5931952 A	☒	Parallel processing integrated circuit tester	713/400
7	US 5930508 A	☒	Method for storing and decoding instructions for a microprocessor having a plurality of function units	717/6
8	US 5880739 A	☒	Blitting of images using instructions	345/433
9	US 5860152 A	☒	Method and apparatus for rapid computation of target addresses for relative control transfer instructions	711/213
10	US 5801676 A	☒	Image display apparatus for processing graphics instructions from a storage device	345/123
11	US 5748642 A	☒	Parallel processing integrated circuit tester	714/724
12	US 5369568 A	☒	Position controlling method of robot	700/61
13	US 5307506 A	☒	High bandwidth multiple computer bus apparatus	710/127
14	US 5179680 A	☒	Instruction storage and cache miss recovery in a high speed multiprocessing parallel processing apparatus	711/125
15	US 5057837 A	☒	Instruction storage method with a compressed format using a mask word	341/55
16	US 5011156 A	☒	Board game apparatus	273/237
17	US 4920477 A	☒	Virtual address table look aside buffer miss recovery method and apparatus	711/207
18	US 4833599 A	☒	Hierarchical priority branch handling for parallel execution in a parallel processor	712/236
19	US 4437149 A	☒	Cache memory architecture with decoding	712/213
20	US 4415767 A	☒	Method and apparatus for speech recognition and reproduction	704/243
21	US 3872442 A	☒	SYSTEM FOR CONVERSION BETWEEN CODED BYTE AND FLOATING POINT FORMAT	708/204
22	US 3772654 A	☒	METHOD AND APPARATUS FOR DATA FORM MODIFICATION	341/60

of tag words at a plurality of tag addresses, each of said plurality of tag words being identical to said first portion of one of said plurality of memory addresses of said main memory, said cache memory having a first address register, the method comprising:

coupling said first address register to a second address register;

coupling said first address register to a first decoder, said first decoder being coupled to said tag memory;

coupling said second address register to a second decoder, said second decoder being coupled to said data memory;

coupling said tag memory to a comparator;

receiving a first memory address in said first address register from said address bus, said first memory address being one of said plurality of said memory addresses;

transmitting said first portion of said first memory address to said comparator;

transmitting said second portion of said first memory address to said first decoder;

transmitting said second portion of said first memory address to said second address register;

decoding said second portion of said first memory address to provide a tag address during said first clock period;

transmitting a first tag word stored in said tag memory at said tag address to said comparator during said first clock period;

comparing, in said comparator, said first tag word with said first portion of said first memory address and outputting a first indication indicating whether said first tag word is identical with said first portion of said first memory address;

decoding the contents of said second address register to provide a first data cache address during said second clock period;

storing data received on said data bus into said data memory at said first data cache address during said second clock period when said first indication indicates that said first tag word is identical with said first portion of said first memory address;

receiving a second memory address in said first address register from said address bus during said second clock period, said first memory address being one of said plurality of said memory addresses;

transmitting said first portion of said second memory address to said comparator;

transmitting said second portion of said second memory address to said first decoder.

8. A cache memory apparatus comprising:

a data memory means for storing a plurality of data words, at least one of said data words being identical to a word stored in a main memory at a store address of said main memory;

a tag memory means for storing tag words, at least one of said tag words containing at least portions of said store address;

a first address register means for storing first information comprising said store address;

a second address register means for storing second information, said second information being at least part of said store address;

path means for transferring said second information to said second address register;

first decoder means for decoding said second information and selecting an entry of said tag memory to provide a selected tag entry;

second decoder means, different from said first decoder means, for decoding the contents of said second address register and selecting an entry of said data memory to provide a selected data entry;

comparing means for comparing third information, comprising tag information stored in said selected tag entry of said tag memory, with fourth information comprising that part of the store address which does not include the store address decoded by said first decoder means;

gate means for permitting writing of information to said data memory at said selected data entry when said comparing means has detected that said third information matches said fourth information; and

means for storing information, different from said first information, in said first address register substantially simultaneously with writing of information to said data memory at said selected data entry.

9. A cache memory apparatus comprising:

a data memory storing a plurality of data information words, at least one of said data information words being identical to a word stored in a main memory at a store address of said main memory,

a tag memory different from said data memory storing tag information words containing at least portions of said store address,

a first address register storing first information comprising said store address, said first address register stores a store address having a 32-bit length,

a path for transferring second information to a second address register, wherein said second information contains at least a portion of said store address and includes six lower order bits of said 32-bit length of said store address,

a first entry decoder, connected to said path for transferring, which decodes said second information and selects an entry of said tag memory to define a selected entry,

a second entry decoder, different from said first entry decoder, which decodes the contents of said second address register, and selects an entry of said data memory, and

a comparator which compares third information comprising tag information stored in the selected entry of said tag memory with fourth information comprising that part of the store address which does not include the store address decoded by the first entry decoder, wherein when an instruction to write information to said main memory is executed, said first address register stores said store address, and when said comparing means has detected that said third information matches said fourth information, said second information is transferred to said second address register through said path.

10. The method of claim **7** further comprising the steps of:

performing a bus snooping operation on said first address register and said first decoder,

wherein said step of performing a bus snooping occurs substantially simultaneously with said step of storing data into said data memory.

* * * * *

	Document	U	Title	Current OR
23	USD 3626427 A	☒	LARGE-SCALE DATA PROCESSING SYSTEM	712/244

US005446876A

United States Patent [19]

Levine et al.

[11] Patent Number: 5,446,876

[45] Date of Patent: Aug. 29, 1995

[54] **HARDWARE MECHANISM FOR INSTRUCTION/DATA ADDRESS TRACING**

[75] Inventors: **Frank E. Levine; Brian C. Twichell; Edward H. Welbon**, all of Austin, Tex.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **228,326**

[22] Filed: **Apr. 15, 1994**

[51] **Int. Cl.**[6] **G06F 13/00**

[52] **U.S. Cl.** **395/184.01**; 364/267.5

[58] **Field of Search** 395/578, 775, 800; 364/232.23, 266.6, 267.4, 267.8, 267.5, 285.3

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,205,370	5/1980	Hirtle .	
4,438,490	3/1984	Wilder, Jr. .	
4,571,677	2/1986	Hirayama et al. .	
4,590,550	5/1986	Eilert et al. .	
4,598,364	7/1986	Gum et al. .	
4,611,281	9/1986	Suko et al. .	
4,636,941	1/1987	Suko .	
4,695,946	9/1987	Andreasen et al. .	
4,783,762	11/1988	Inoue et al. .	
4,802,165	1/1989	Ream .	
4,879,646	11/1989	Iwasaki et al. .	
5,073,968	12/1991	Morrison .	
5,127,103	6/1992	Hill et al. .	
5,134,701	7/1992	Mueller et al. .	
5,146,586	9/1992	Nakano .	
5,182,811	1/1993	Sakamura	395/800
5,220,669	6/1993	Baum et al. .	

FOREIGN PATENT DOCUMENTS

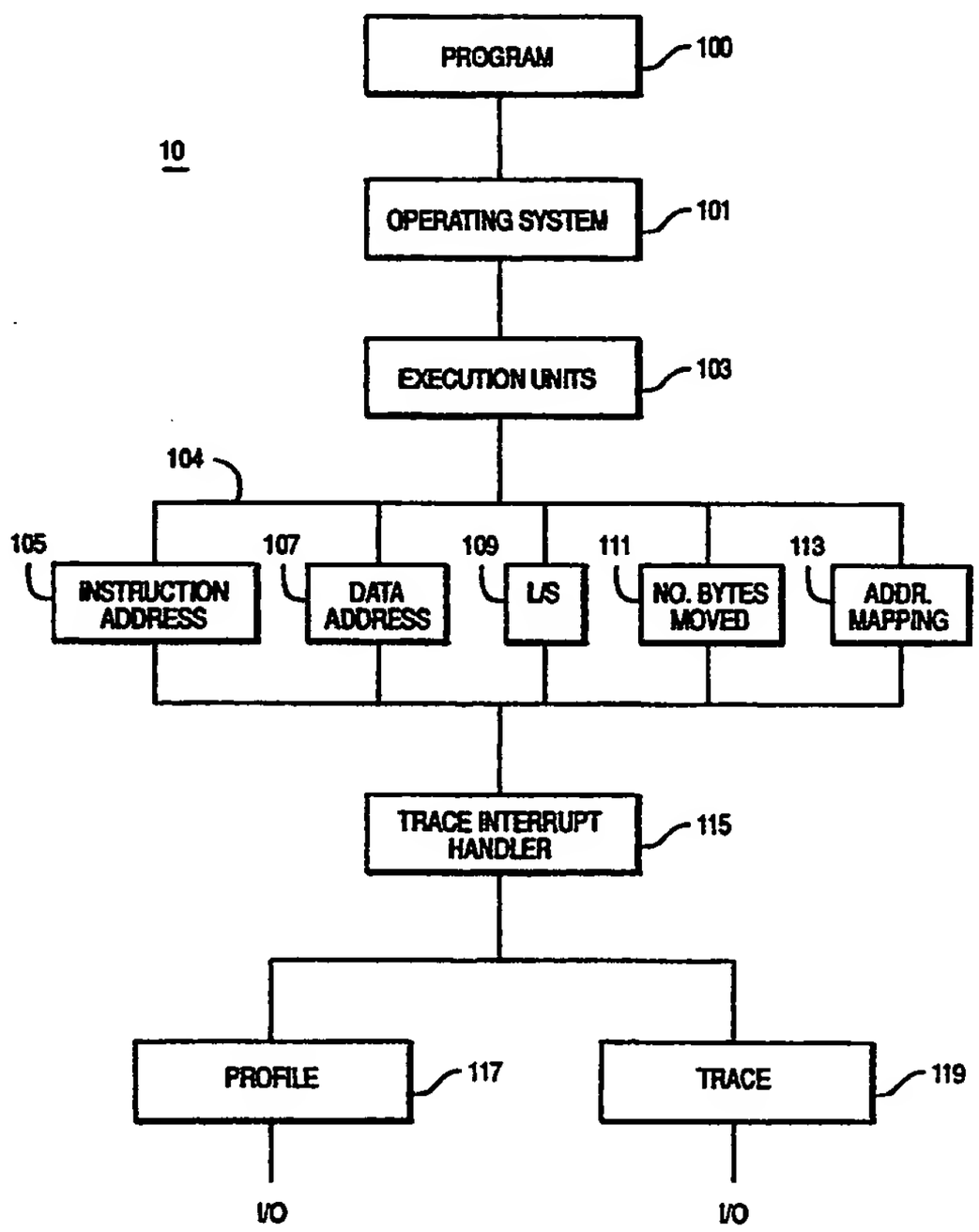
0270983A2	6/1988	European Pat. Off. .
0501076A2	9/1992	European Pat. Off. .
0525672A2	2/1993	European Pat. Off. .

Primary Examiner—Vincent P. Canney
Attorney, Agent, or Firm—Mark E. McBurney

[57] **ABSTRACT**

An improved instruction tracing mechanism provides a combination of hardware, internal to the CPU, and novel software. Additional registers are added to interconnected to the CPU. These registers store values indicating the instruction address, data address, whether the instruction was a load or store, the number of bytes moved and whether any address mapping changes occurred. The registers are read by a trace interrupt handler which then provides the information to a trace buffer and a profile buffer. The end user can then access the trace and profile information through the input/output (I/O) system of the data processing system.

16 Claims, 6 Drawing Sheets



	Document	U	Title	Current OR
1	USD 61382 54 A	☐	Method and apparatus for redundant location addressing using data compression	714/710
2	US 61311 52 A	☒	Planar cache layout and instruction stream therefor	712/24
3	US 61280 94 A	☒	Printer having processor with instruction cache and compressed program store	358/1.15
4	US 61157 39 A	☒	Image scanner adapted for direct connection to client/server type network	709/215
5	US 61087 72 A	☒	Method and apparatus for supporting multiple floating point processing models	712/221
6	US 61009 05 A	☒	Expansion of data	345/501
7	US 60498 62 A	☒	Signal processor executing compressed instructions that are decoded using either a programmable or hardwired decoder based on a category bit in the instruction	712/208
8	US 60444 50 A	☒	Processor for VLIW instruction	712/24
9	US 60285 90 A	☒	Color conversion for processors	345/154
10	US 60115 79 A	☒	Apparatus, method and system for wireline audio and video conferencing and telephony, with network interactivity	348/15
11	US 60095 08 A	☒	System and method for addressing plurality of data values with a single address in a multi-value store on FIFO basis	712/41
12	US 59742 20 A	☒	Video editing method, non-linear video editing apparatus, and video editing program storage medium	386/52
13	US 59564 66 A	☒	Image processor capable of operation as a facsimile	358/1.9
14	US 59432 02 A	☒	Two way packet radio including smart data buffer and packet rate conversion	361/66
15	US 59352 56 A	☒	Parallel processing integrated circuit tester	713/400
16	US 59319 53 A	☒	Parallel processing integrated circuit tester	713/500
17	US 59319 52 A	☒	Parallel processing integrated circuit tester	713/400
18	US 59335 80 A	☒	Scanner printer server	358/1.13
19	US 59305 08 A	☒	Method for storing and decoding instructions for a microprocessor having a plurality of function units	717/6
20	US 59268 16 A	☒	Database Synchronizer	707/8
21	US 59206 51 A	☒	Compressed data expanding apparatus	382/233

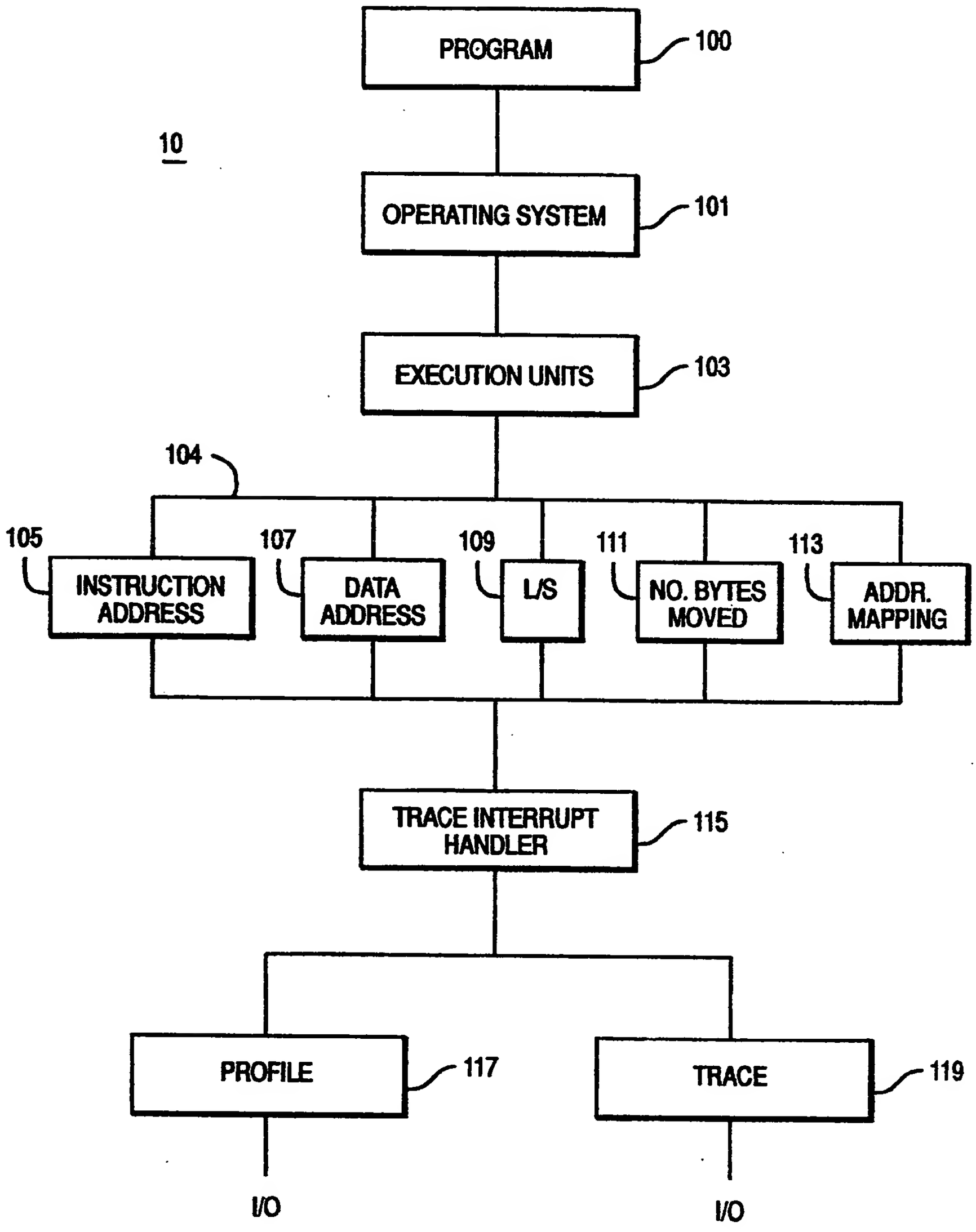

10
PROGRAM
100
OPERATING SYSTEM
101
EXECUTION UNITS
103
104
105
INSTRUCTION ADDRESS
107
DATA ADDRESS
109
L/S
111
NO. BYTES MOVED
113
ADDR. MAPPING
TRACE INTERRUPT HANDLER
115
PROFILE
117
TRACE
119
I/O
I/O

FIG. 2

	Document	U	Title	Current OR
22	US 5905893 A	☒	Microprocessor adapted for executing both a non-compressed fixed length instruction set and a compressed variable length instruction set	717/5
23	US 5898462 A	☒	Methods of producing data storage devices for appliances which can be used to coach users in the performance of user-selected tasks	348/552
24	US 5896519 A	☒	Apparatus for detecting instructions from a variable-length compressed instruction set having extended and non-extended instructions	712/213
25	US 5884325 A	☒	System for synchronizing shared data between computers	707/201
26	US 5881260 A	☒	Method and apparatus for sequencing and decoding variable length instructions with an instruction boundary marker within each instruction	712/210
27	US 5878267 A	☒	Compressed instruction format for use in a VLIW processor and processor for processing such instructions	712/24
28	US 5870765 A	☒	Database synchronizer	707/203
29	US 5870759 A	☒	System for synchronizing data between computers using a before-image of data	707/201
30	US 5870576 A	☒	Method and apparatus for storing and expanding variable-length program instructions upon detection of a miss condition within an instruction cache containing pointers to compressed instructions for wide instruction word processor architectures	712/210
31	US 5867681 A	☒	Microprocessor having register dependent immediate decompression	712/208
32	US 5862398 A	☒	Compiler generating swizzled instructions usable in a simplified cache layout	712/24
33	US 5852741 A	☒	VLIW processor which processes compressed instruction format	712/24
34	US 5835749 A	☒	Method and apparatus for providing dynamically linked libraries	709/331
35	US 5826054 A	☒	Compressed Instruction format for use in a VLIW processor	712/213
36	US 5822507 A	☒	Scanner printer server for selectively controlling a scanner and a printer in accordance with a copy operation	358/1.15
37	US 5822493 A	☒	Real-time image recording/producing method and apparatus and video library system	386/109
38	US 5819058 A	☒	Instruction compression and decompression system and method for a processor	712/210
39	US 5801784 A	☒	Data storage devices	348/552
40	US 5794010 A	☒	Method and apparatus for allowing execution of both compressed instructions and decompressed instructions in a microprocessor	703/20
41	US 5790856 A	☒	Methods, apparatus, and data structures for data driven computer patches and static analysis of same	717/3

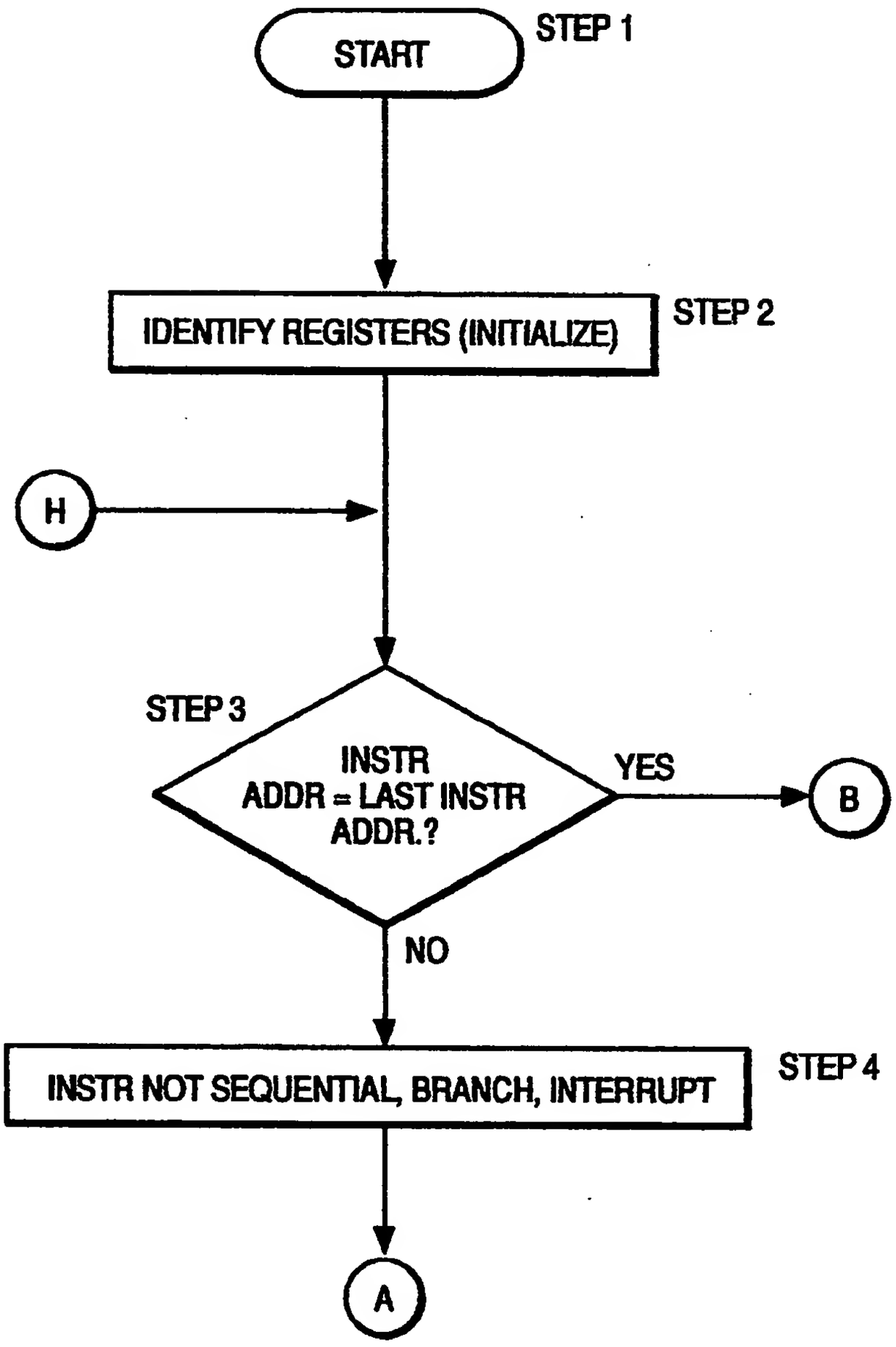

START
STEP 1
IDENTIFY REGISTERS (INITIALIZE)
STEP 2
H
STEP 3
INSTR
ADDR = LAST INSTR
ADDR.?
YES
B
NO
INSTR NOT SEQUENTIAL, BRANCH, INTERRUPT
STEP 4
A

FIG. 3A

	Document	U	Title	Current OR
42	USD 57873 02 A	☒	Software for producing instructions in a compressed format for a VLIW processor	712/24
43	US 57845 85 A	☒	Computer system for executing instruction stream containing mixed compressed and uncompressed instructions by automatically detecting and expanding compressed instructions	712/209
44	US 57643 04 A	☒	Operation of information/entertainment centers	348/552
45	US 57486 42 A	☒	Parallel processing integrated circuit tester	714/724
46	US 57200 13 A	☒	Scanner printer server and method for selectively outputting scanned information to an information processing apparatus in accordance with a pre-scan command and a scan command	358/1.15
47	US 56995 36 A	☒	Computer processing system employing dynamic instruction formatting	712/216
48	US 56943 99 A	☒	Processing unit for generating signals for communication with a test access port	709/246
49	US 56662 16 A	☒	Image processing apparatus and method	358/500
50	US 56641 63 A	☒	Image generating method and apparatus	345/522
51	US 56528 52 A	☒	Processor for discriminating between compressed and non-compressed program code, with prefetching, decoding and execution of compressed code in parallel with the decoding, with modified target branch addresses accommodated at run time	712/208
52	US 56469 46 A	☒	Apparatus and method for selectively companding data on a slot-by-slot basis	370/442
53	US 56320 24 A	☒	Microcomputer executing compressed program and generating compressed branch addresses	712/205
54	US 56196 98 A	☒	Method and apparatus for patching operating systems	717/10
55	US 55686 50 A	☒	Control unit for controlling reading and writing of a magnetic tape unit	710/52
56	US 55637 19 A	☒	Data recording/replay device and data recording medium	358/436
57	US 55173 31 A	☒	Method and apparatus for reading image of image scanner-reader	358/486
58	US 55112 10 A	☒	Vector processing device using address data and mask information to generate signal that indicates which addresses are to be accessed from the main memory	712/5
59	US 54423 50 A	☒	Method and means providing static dictionary structures for compressing character data and expanding compressed data	341/51
60	US 54325 32 A	☒	Video printer for printing plurality of kinds of images of different image formats	347/176

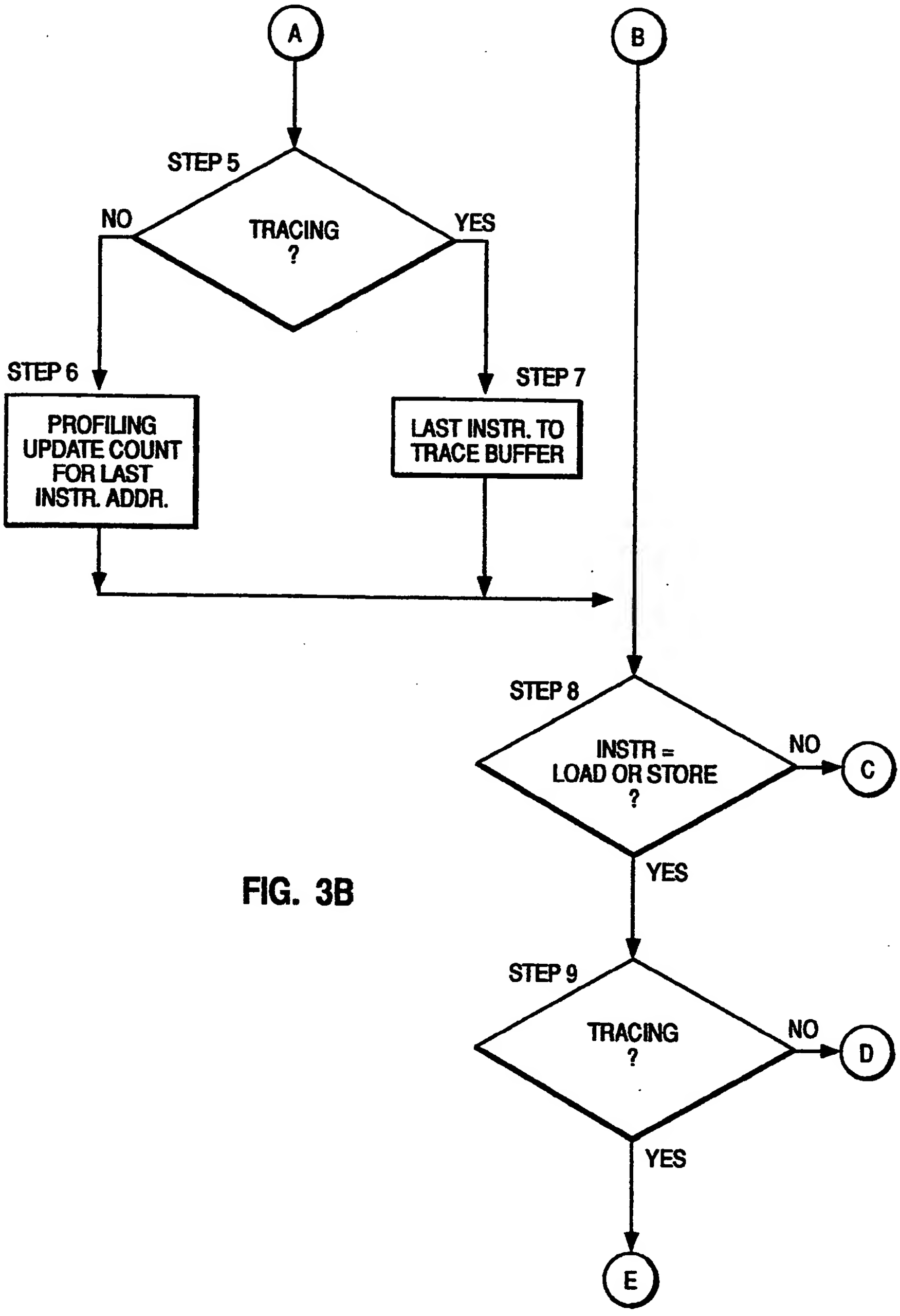

A
STEP 5
NO
TRACING ?
YES
STEP 6
PROFILING UPDATE COUNT FOR LAST INSTR. ADDR.
STEP 7
LAST INSTR. TO TRACE BUFFER
B
STEP 8
INSTR = LOAD OR STORE ?
NO
C
YES
STEP 9
TRACING ?
NO
D
YES
E

FIG. 3B

	Document	U	Title	Current OR
61	US 5419146 A	☒	Evaporator water temperature control for a chiller system	62/115
62	US 5321619 A	☒	Production control method and system therefor	700/116
63	US 5321232 A	☒	Oven controlled by an optical code reader	219/506
64	US 5317604 A	☒	Isochronous interface method	375/240
65	US 5317603 A	☒	Isochronous interface apparatus	375/240
66	US 5263335 A	☒	Operation controller for air conditioner	62/228.4
67	US 5245676 A	☒	Determination of image skew angle from data including data in compressed form	382/235
68	US 5170445 A	☒	Document decompressing system	382/233
69	US 5046108 A	☒	Imaging processing method and apparatus suitably used for obtaining shading image	382/131
70	US 5021892 A	☒	Image processing device of multifunctional type	358/468
71	US 4918624 A	☒	Vector generator scan converter	358/1.15
72	US 4910607 A	☒	Image processing device of multifunctional type	358/400
73	US 4873630 A	☒	Scientific processor to support a host processor referencing common memory	712/3
74	US 4715191 A	☒	Air conditioning method	62/208
75	US 4528640 A	☒	Method and a means for checking normalizing operations in a computer device	708/530
76	US 4458110 A	☒	Storage element for speech synthesizer	704/211
77	US 4384170 A	☒	Method and apparatus for speech synthesizing	704/266
78	US 4384169 A	☒	Method and apparatus for speech synthesizing	704/206
79	US 4373349 A	☒	Heat pump system adaptive defrost control system	62/156
80	US 4214125 A	☒	Method and apparatus for speech synthesizing	704/268
81	US 4099205 A	☒	Phase control system	348/513
82	US 3885207 A	☒	Optimized editing system for a servo controlled program recording system	318/568.14

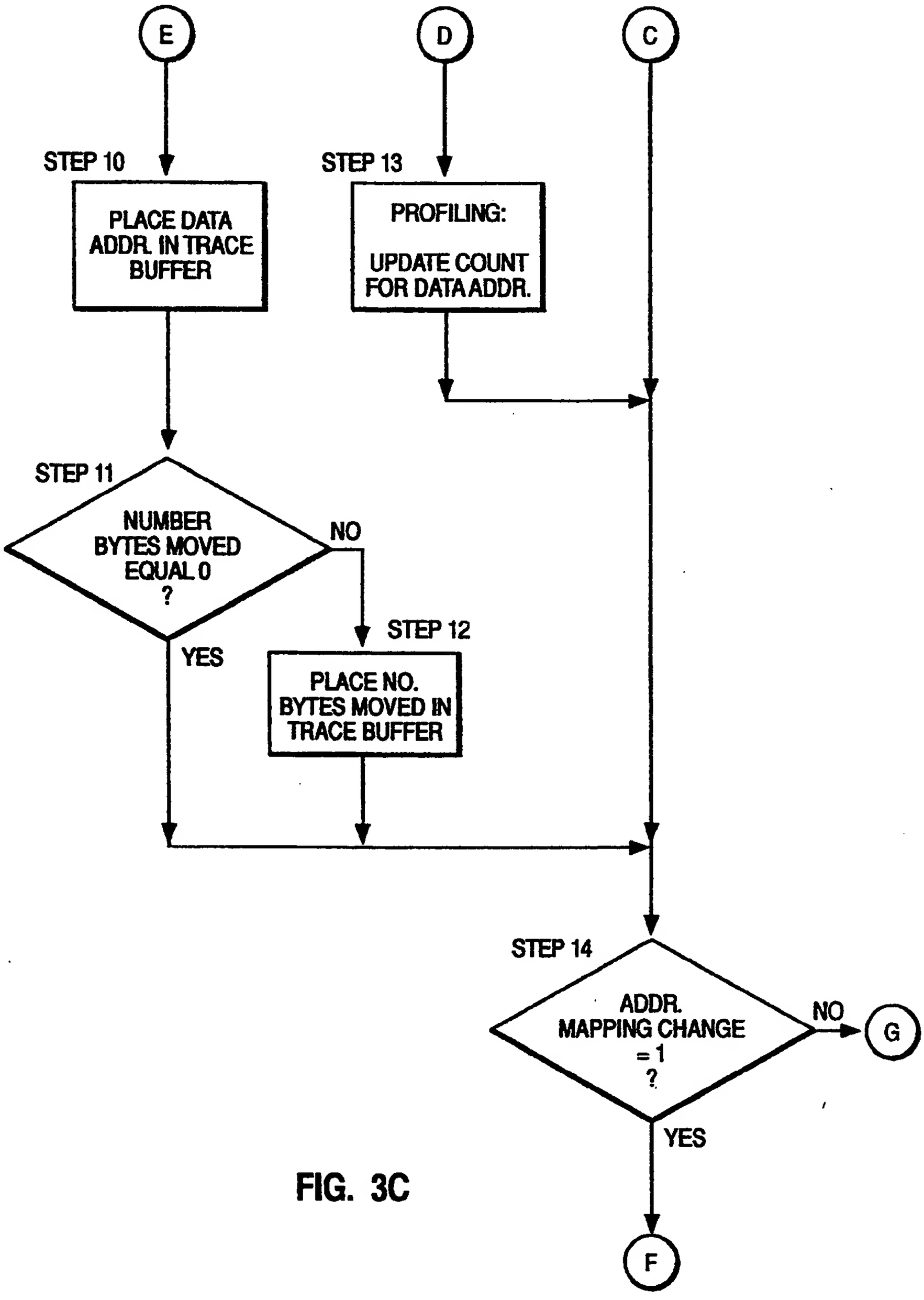

E
STEP 10
PLACE DATA ADDR. IN TRACE BUFFER
D
STEP 13
PROFILING:
UPDATE COUNT FOR DATA ADDR.
C
STEP 11
NUMBER BYTES MOVED EQUAL 0 ?
NO
YES
STEP 12
PLACE NO. BYTES MOVED IN TRACE BUFFER
STEP 14
ADDR. MAPPING CHANGE =1 ?
NO
G
YES
F

FIG. 3C

	Document	U	Title	Current OR
83	USD 38724 42 A	☒	SYSTEM FOR CONVERSION BETWEEN CODED BYTE AND FLOATING POINT FORMAT	708/204

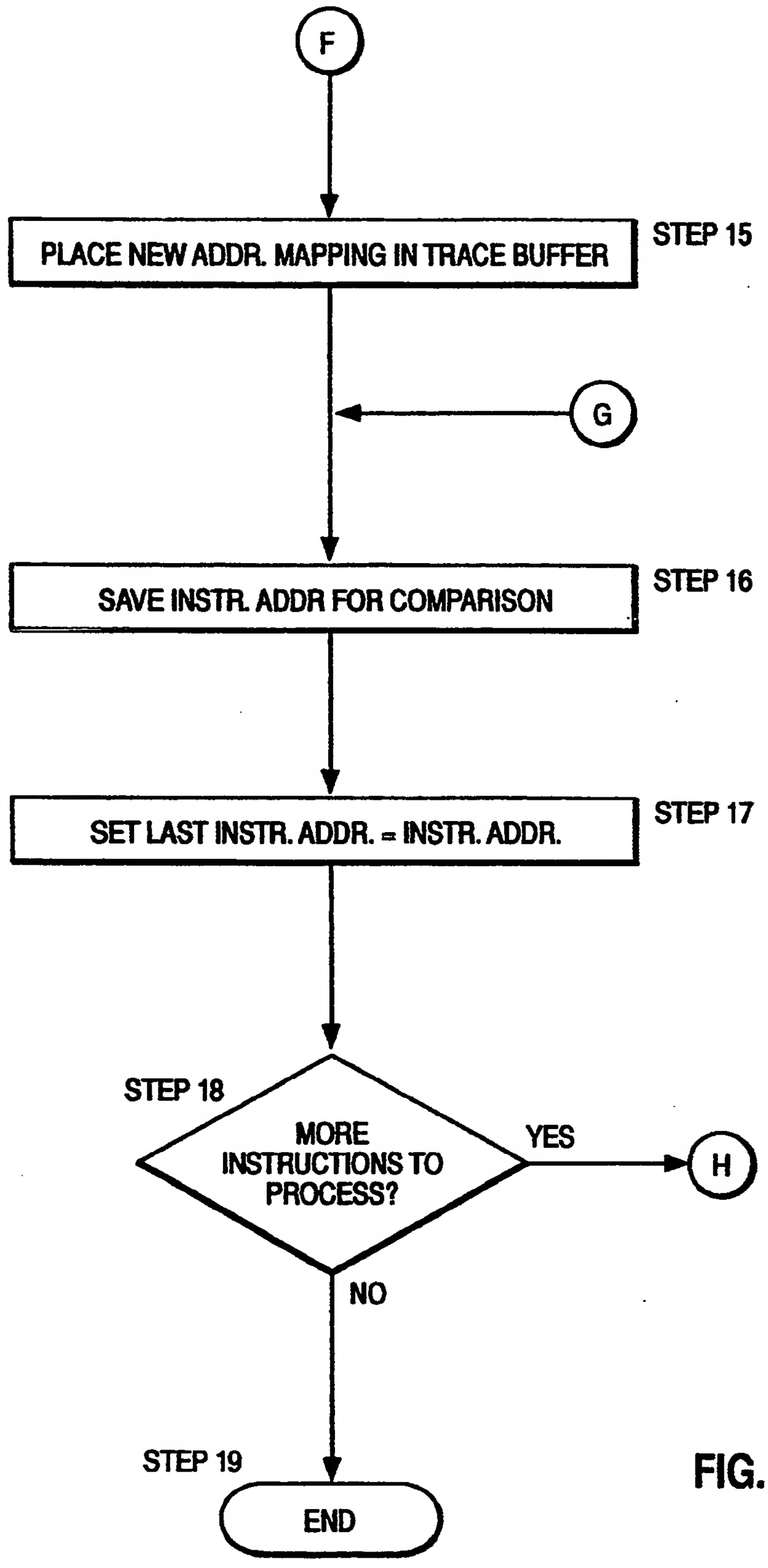

F
PLACE NEW ADDR. MAPPING IN TRACE BUFFER
STEP 15
G
SAVE INSTR. ADDR FOR COMPARISON
STEP 16
SET LAST INSTR. ADDR. = INSTR. ADDR.
STEP 17
STEP 18
MORE INSTRUCTIONS TO PROCESS?
YES
H
NO
STEP 19
END

FIG. 3D

	Document	U	Title	Current OR
1	USD 61380 39 A	☐	Communication terminal apparatus and control method thereof	455/566
2	US 61311 62 A	☒	Digital data authentication method	713/176
3	US 61310 48 A	☒	Communication terminal apparatus and control method thereof	455/566
4	US 61287 14 A	☒	Method of processing a data move instruction for moving data between main storage and extended storage and data move instruction processing	711/165
5	US 60960 89 A	☒	Power simulation system, power simulation method and computer-readable recording medium for recording power simulation program	703/18
6	US 60442 59 A	☒	System for subscriber administration in telecommunication network	455/406
7	US 60397 65 A	☒	Computer instruction which generates multiple results of different data types to improve software emulation	703/26
8	US 60413 51 A	☒	Network traffic by instruction packet size reduction	709/224
9	US 60292 28 A	☒	Data prefetching of a load target buffer for post-branch instructions based on past prediction accuracy's of branch predictions	711/137
10	US 60092 58 A	☒	Methods and devices for unwinding stack of frozen program and for restarting the program from unwound state	703/22
11	US 59998 27 A	☒	Communication terminal apparatus and control method thereof	455/564
12	US 59873 36 A	☒	Communication terminal apparatus and control method thereof	455/566
13	US 59876 16 A	☒	Semiconductor device	713/320
14	US 59833 59 A	☒	Processor fault recovering method for information processing system	714/10
15	US 59745 38 A	☒	Method and apparatus for annotating operands in a computer system with source instruction identifiers	712/218
16	US 59616 32 A	☒	Microprocessor with circuits, systems, and methods for selecting alternative pipeline instruction paths based on instruction leading codes	712/212
17	US 59580 43 A	☒	Superscalar processor with forward map buffer in multiple instruction parallel issue/execution management system	712/216
18	US 59516 79 A	☒	Microprocessor circuits, systems, and methods for issuing successive iterations of a short backward branch loop in a single cycle	712/241
19	US 59448 11 A	☒	Superscalar processor with parallel issue and execution device having forward map of operand and instruction dependencies	712/23
20	US 59387 59 A	☒	Processor instruction control mechanism capable of decoding register instructions and immediate instructions with simple configuration	712/209

mation to a trace buffer and a profile buffer. The end user can then access the trace and profile information through the input/output (I/O) system of the data processing system. That is, the user can read the information from the display, store it to a disk, or the like.

The trace facility of the present invention allows improved performance when compared to prior art software and external hardware systems, and provides the additional information of the address from which the data is retrieved, or stored.

These and other objects, features and advantages will become apparent to those skilled in the art upon considering the subsequent description taken in conjunction with the attached drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block of a data processing having a central processing unit including the trace facility of the present invention;

FIG. 2 is a block diagram of the hardware aspect of the present invention that is included in the central processing unit; and

FIG. 3 consisting of 3A–3D is a flow chart showing the process steps implemented by the software aspect of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Generally, tracing tools are used to evaluate program applications running on a particular computer system. In particular the instructions to be executed by the CPU are analyzed for their content. Another important aspect of tracing is to determine where the operand data to be manipulated by the instruction is stored. That is, what is the address of the data to be operated on by the instruction? This information will allow a system designer to implement methods that will improve the performance of the software.. These methods include improvements to the software itself, and improvements to future hardware systems.

More specifically, a program to be evaluated is run in conjunction with a testing tool, such as a tracing tool, or the like. This tracing tool will output two basic types of information. First, profile data that is essentially a set of counts corresponding to the instructions executed by the CPU. For example, the profile data may include the number of instructions that were executed by a given software routine. This information is then used by the system designer to optimize the performance of the software and/or hardware running the program. That is, the designer will look at the number of instructions executed per routine and try to minimize these executions to make the program run more efficiently. Additionally, the profile data will provide information regarding code coverage of the trace, i.e. how many lines of code (LOC) executed during the test. This information will tell the designer how reliable the trace test was. The more LOC that were executed, the higher the test coverage and the better the test.

Second, a trace tool will output trace information which is time sequences of events that occurred on the CPU. More particularly, the trace information includes instructions that executed on the processor and when these instructions executed. One common used of trace information is in the area of hardware simulation. Often computer hardware designers will create a software model of a chip being designed. Before the design is actually committed to fabrication, the traces information can be used as input to the model and the output monitored to determine how well a proposed hardware design will run the program corresponding to the trace input. This is a great advantage, since hardware designers have the capability of knowing, before the chip(s) are even fabricated, how well certain programs will operate on the designed system.

Trace information is also used for performance debugging to look at where the program stores and retrieves operands, when the CPU executed the instructions, and the like. This will allow the hardware designers to understand how their system works when running a particular program. It also gives the software designers more information about how their program operates and lets them optimize its performance. To improve both the robustness and performance of the trace facility of the present invention, the following hardware support has been provided in the CPU. The central processing unit may be one of the PowerPC microprocessors available from IBM Corp. (PowerPC is a trademark of IBM Corp.). The CPU of the present invention includes a single step interrupt capability that vectors uniquely to a trace exception handler before the execution of each next instruction. Additionally, special registers, the Saved Instruction Address (SIA) and Saved Data Address (SDA) are provided that will contain the effective address of the last architecturally successfully executed instruction and the instruction's operand. A bit field that can be tested to determine if the last architecturally successfully executed instruction was a load or store is provided such that these bits are used to validate the content of the SDA.

A bit field that can be tested to determine if the last architecturally successfully executed instruction was a load or store multiple is provided. These bits are used to determine if the number of memory elements accessed by the load or store was dynamically determined and not easily obtained via post processing. Another bit field is provided that can be tested to determine if the last architecturally successfully executed instruction altered the effective to virtual address map. This results in considerable savings to the trace overhead since conventional current software testing requires that addresses in machine code be .correlated with source code (i.e. the corresponding high level language statements that generate the machine code). Further, a control mechanism is provided that either allows or disallows the assertion of the trace interrupt before execution of each instruction.

The present invention utilizes a software trace algorithm in conjunction with hardware additions to the CPU to efficiently provide instruction profiles and traces in conjunction with data address which the instructions operated on. When trace is enabled, a trace exception interrupt is taken before each instruction execution. Once the trace exception handler receives control, a pointer will indicate the address of the next instruction to be executed. As the last step of the prior trace interrupt, the previous instruction address will have been saved in a memory location (Last_Instruction_Address). Thus, the memory location will contain the .address of the last architecturally successfully executed instruction. If the content of memory location with the last data address represents a break in flow of the traced instruction stream (e.g. branch etc.) then the contents of that memory location are written to the trace medium (typically a buffer in memory).

	Document	U	Title	Current OR
21	US 5934020 A	☒	Window lock and guard	49/56
22	US 5930799 A	☒	Database creating method using image information	707/102
23	US 5926639 A	☒	Embedded flow information for binary manipulation	717/5
24	US 5909565 A	☒	Microprocessor system which efficiently shares register data between a main processor and a coprocessor	712/200
25	US 5897633 A	☒	System for converting programs and databases to correct year 2000 processing errors	707/6
26	US 5898704 A	☒	Processing system having testing mechanism	714/727
27	US 5896521 A	☒	Processor synthesis system and processor synthesis method	703/21
28	US 5881277 A	☒	Pipelined microprocessor with branch misprediction cache circuits, systems and methods	712/239
29	US 5867498 A	☒	Intelligent telecommunications network	370/385
30	US 5860156 A	☒	Method for implementing an indexed jump table	711/221
31	US 5841051 A	☒	Apparatus for providing musical instruction	84/477R
32	US 5841792 A	☒	Processing system having a testing mechanism	714/733
33	US 5828860 A	☒	Data processing device equipped with cache memory and a storage unit for storing data between a main storage or CPU cache memory	712/207
34	US 5809500 A	☒	System for converting programs and databases to correct year 2000 processing errors	707/6
35	US 5799180 A	☒	Microprocessor circuits, systems, and methods passing intermediate instructions between a short forward conditional branch instruction and target instruction through pipeline, then suppressing results if branch taken	712/234
36	US 5787303 A	☒	Digital computer system capable of processing a plurality of instructions in parallel based on a VLIW architecture	712/24
37	US 5784607 A	☒	Apparatus and method for exception handling during micro code string instructions	712/245
38	US 5774711 A	☒	Apparatus and method for processing exceptions during execution of string instructions	712/244
39	US 5752015 A	☒	Method and apparatus for repetitive execution of string instructions without branch or loop microinstructions	712/241
40	US 5745740 A	☒	Parallel processing computer system, constituent units for use therein, and clock tuning method for a parallel processing computer system	713/400

If the last architecturally successfully executed instruction was a load or store then the address of the operand will be contained in the SDA and will be written to the trace medium. A bit field, as described above, is then used to make the required determination as to whether the instruction was a load or store. If the number of items accessed was dynamically determined, then the actual number must be noted on the trace medium. Another bit field, also as described above, is then used to make the required determination as to whether the instruction was a load multiple or a store multiple. If the addressability (effective to virtual address mapping) has been changed then the aspect that changed (e.g. BAT or SEG registers) must be noted on the trace medium. The bit field that determines if the executed instruction altered the effective to virtual address map is used to make the required determination. After utilizing the content of the address of the last instruction, the content of that memory location is replaced by the content of the current instruction address in anticipation of the next trace handler invocation. Thus, sequential instruction streams are traced.

There are two major performance improvements. The first is that the effort expended in computing the last instructions operand address and length has been reduced to testing bit fields (to determine if the last instruction executed was a load or store, or a load or store multiple, respectively). These bit fields along with the SDA register eliminate the need for instrumentation of the loads and stores in the traced code.

The second performance improvement is that detecting effective to virtual address mapping changes is very efficient, being reduced to testing the bit field that determines if the effective to virtual address mapping has been altered. Since the overall rate of addressability changes is typically low, knowing when addressability has not changed allows addressability searches to be performed only when required thus benefiting performance.

A significant improvement in the robustness of the software tracing aspect of the present invention comes about as consequence of the simplifications afforded the tracing software by the ability to determine when the effective to virtual addressing has changed. Also, the addition of the ability to efficiently determine the operand addresses and lengths makes the mechanism which eliminates the need to manage the tracing of various system components more operable.

The current estimate for the trace handler of the present invention places the path length at 15 instructions when there is no write to the trace medium, e.g. trace buffer, and 20 instructions when there is a write to the trace medium. Assuming a basic block length of 5 instructions, the percentage of load and stores in the traced instructions stream to be 40% and the percentage of taken branches to be 5%, leads to a mean interrupt handler path length of 17.25 instructions.

Assuming an interrupt latency of 10 cycles and an interrupt Cycles Per Instruction (CPI) figure of 0.8 the mean number of cycles added to the execution time of each instruction of the workload will be about 23.8 cycles. Since most processor instructions contemplated by the present invention can execute in one cycle, the effective CPI for the traced workload would be about 24.8 cycles per instruction.

Note that there is no need to disable the caches in software tracing. The least recently used (LRU) replacement strategy of the caches essentially insures that when software tracing, the trace code will be in the cache, thus the trace handler CPI may be less than 0.8 and consequently execute in less that 24.8 cycles. Also, since the trace interrupt is a special case, the present invention can take into account the special conditions surrounding a trace interrupt. Consequently, it is reasonable to believe that it is possible to obtain very attractive trace execution rates.

A performance contrast exists between the present invention, which utilizes both a novel hardware scheme and a new software portion, to that of the typical cache inhibited external hardware trace scheme. In machines that depend on low cache miss rates, the cache inhibited scheme may result in a severe penalty, i.e. the more code parallelism a processor can exploit, the more critical the cache miss ratio will be to the processor's performance.

Assuming the following simple linear model of execution:

$$\text{CPI(Miss\_Ration)} = \text{CPI}_{infinity} + \text{Miss\_Ratio X Miss\_Cost}$$

The degree to which the processor is slowed down by hardware tracing that disables the cache is:

$$\text{Trace\_Slow\_Down} = \frac{CPI(\text{Tracing Miss Ration} = 100\%)}{CPI(\text{Normal\_Miss\_Ratio})}$$

Applying the linear mode; to the last equation gives the following:

$$\text{Trace\_Slow\_Down} = \frac{1 + \frac{\text{Miss Cost}}{CPI_{infinity}}}{1 + \text{Normal\_Miss\_Ratio} \times \frac{\text{Miss Cost}}{CPI_{infinity}}}$$

Assuming that the code being traced has an infinite cache CPI of 1.6, the cost of a miss is 16 cycles and the normal miss ratio is 1% then the trace slow down is about 10. On the other hand, if the code being traced has an infinite cache CPI of 0.25, the cost of a miss is 16 cycles and the normal miss ratio is 0.5% then the trace slow down is about 49.1.

This last value corresponds to an effective workload CPI while hardware tracing of 12.3 cycles per instruction. Assuming that a processor capable of sustaining a workload CPI of 0.25 when not tracing can also provide a trace handler CPI of 0.5, the effective workload CPI while software tracing would be 19.5 cycles per instruction or about 58% slower than hardware tracing.

The above computations are examples intended to show that in a real processor the slow down imposed by external hardware tracing can actually turn out to be comparable to that of software tracing, in other words, there is no real advantage of external hardware tracing over software tracing. Thus, the present invention becomes very attractive when compared to the prior art systems. This is particularly true for machines that exploit code parallelism and multiple storage hierarchies.

Other limitations such as electrical loading of high speed signals by trace equipment could force the traced system to be operated at a reduced clock frequency resulting in a further decrease of the speed of hardware tracing. Also, most RISC processors have external busses that run at a fraction of the speed of the processor which accentuates the bandwidth requirements of hardware tracing. Such problems can only worsen as ma-

	Document	U	Title	Current OR
41	USD 57449 49 A	☒	Analog test cell circuit	324/158.1
42	US 57427 83 A	☒	System for grouping instructions for multiple issue using plural decoders having forward and backward propagation of decoding information	712/212
43	US 57375 79 A	☒	System and method for emulating computer architectures	703/26
44	US 57245 05 A	☒	Apparatus and method for real-time program monitoring via a serial interface	714/45
45	US 57218 65 A	☒	Information processing apparatus with prefetch control for prefetching data structure from memory through cache memory	711/137
46	US 56849 83 A	☒	Microprocessor executing multiple register transfer operations with a single instruction with derivation of destination register numbers from source register	712/225
47	US 56805 68 A	☒	Instruction format with sequentially performable operand address extension modification	711/220
48	US 56689 47 A	☒	Microprocessor self-test apparatus and method	714/30
49	US 56661 39 A	☒	Pen-based computer copy editing apparatus and method for manuscripts	345/173
50	US 56597 85 A	☒	Array processor communication architecture with broadcast processor instructions	712/11
51	US 56491 35 A	☒	Parallel processing system and method using surrogate instructions	712/200
52	US 56340 84 A	☒	Abbreviation and acronym/initialism expansion procedures for a text to speech reader	704/260
53	US 55863 23 A	☒	Compilier system using an intermediate abstract form and machine-specific installers	717/5
54	US 55748 78 A	☒	Method of parallel purging of translation lookaside buffer in a multilevel virtual machine system	711/207
55	US 55487 66 A	☒	Microprocessor for selecting data bus terminal width regardless of data transfer mode	710/127
56	US 55485 73 A	☒	Optical information reproducing apparatus provided with laser power control means for detecting reflected light from data region	369/116
57	US 55349 74 A	☒	Printing apparatus performing bidirectional communication with a plurality of user terminals	399/1
58	US 55111 73 A	☒	Programmable logic array and data processing unit using the same	712/248
59	US 55091 37 A	☒	Store processing method in a pipelined cache memory	711/168
60	US 55069 80 A	☒	Method and apparatus for parallel processing of a large data array utilizing a shared auxiliary memory	711/159

chine clock speeds become higher and packaging becomes more dense. It is inevitable that hardware tracing will continue to become increasingly less tractable and software tracing increasingly more tractable.

Further, the present invention includes a register denoted the "Saved Instruction Address" or SIA. The SDA is an important aspect of the present invention since it will eliminate unnecessary instructions in the trace handler. As described above, reducing path length reflects directly in the performance of the trace.

With the disclosed hardware and software features of the present invention, useful traces can be inexpensively captured with reasonable performance degradation from a standard machine (i.e., a machine not dedicated to tracing). A distinct advantage of this is that stand alone data processing systems including the processor of the present invention can be used to trace instructions, without having either an external hardware tracing device, or special purpose tracing software, thus, providing more utility to the end user.

Referring to FIG. 1, a typical data processing system is shown which may be used in conjunction with the present invention. A central processing unit (CPU), such as the PowerPC 604 microprocessor (PowerPC is a trademark of IBM) commercially .available from IBM is provided and interconnected to the various other components by system bus 12. It should be noted that the hardware trace mechanism of the present invention is part of the function which is included in CPU 10. Read only memory (ROM) 16 is connected to CPU 10 via bus 12 and includes the basic input/output system (BIOS) that controls the basic computer functions. Random access memory (RAM) 14, I/O adapter 18 and communications adapter 34 are also interconnected to system bus 12. Expanded memory 15 is additional RAM added to the data processing system and is also shown interconnected to bus 12. I/O adapter 18 may be a small computer system interface (SCSI) adapter that communicates with a disk storage device 20. Communications adapter 34 interconnects bus 12 with an outside network enabling the data processing system to communicate with other such systems. Input/Output devices are also connected to system bus 12 via user interface adapter 22 and display adapter 36. Keyboard 24, track ball 32, mouse 26 and speaker 28 are all interconnected to bus 12 via user interface adapter 22. Display monitor 38 is connected to system bus 12 by display adapter 36. In this manner, a user is capable of inputting to the system through the keyboard 24, trackball 32 or mouse 26 and receiving output from the system via speaker 28 and display 38.. Additionally, an operating system such as DOS or the OS/2 system (OS/2 is a Trademark of IBM Corporation) is used to coordinate the functions of the various components shown in FIG. 1.

Referring to FIG. 2, a CPU is shown including the tracing hardware of the present invention. Reference 10 refers generally a CPU as is shown in FIG. 1. A computer software program 100 is installed and running in conjunction with an operating system 101, which may be one of AIX, DOS, OS/2, or the like (AIX and OS/2 are trademarks of IBM Corp.). The operating system 101 controls the basic functions of the computer system.

In a preferred embodiment of the present invention, CPU 10 will be one of the IBM reduced instruction set computer (RISC), processing systems as implemented in the POWER and PowerPC Architecture (POWER and PowerPC are trademarks of IBM Corp.). Execution unit 103 will include both a fixed point unit (FXU) and a floating point unit (FPU). The FXU being used primarily for integer processing, while the FPU is normally used for processing number in scientific notation. Both of these execution units receive instructions from an instruction cache unit. The FPU and FXU output information to a bus 104 which is connected to a plurality of register files. Register 105 will contain the instruction address, and register 107 will store the data address (address from which the data is retrieved during a load operation, and the address to which the data is being stored for a store operation). The Load/Store register 109 will contain a binary value that will indicate if either a load or store instruction was executed, by any of the execution units in CPU 10. The Number_of_Bytes_Moved register 111 will include a value representative of the number of bytes of data moved to the memory location during a load or store instruction (if a load or store instruction was actually executed, as indicated by the value in register 109). It should be noted that the term "memory location" as used herein will include all of the components of a hierarchical memory subsystem, including a level one (L1) cache, level two (L2) cache, instruction cache unit, and the like.

Register 113 will contain a specific binary value indicating whether or not the effective to virtual address mapping scheme was changed by the previous instruction. This information is very important in a tracing context, since it may be necessary to use the physical address (data or instruction) to recreate the actual instruction that was executed, or data that was accessed. A mapping table, or look-up table is used to map a 32 bit effective address into a 52 bit virtual address. The actual physical address of the data or instruction is then determined from the virtual address. Periodically, instructions alter, or change, the content of the look-up table such that a particular effective address maps to a different virtual address. It is imperative that the tracing tool log the changes to the table occur when they occur. Otherwise, the actual data or instruction at a specified address may not be obtained by working backwards from the physical address to the virtual address to the effective address.

Trace interrupt handler 115 is a routine contained in the software operating system. The process implemented by this program will be more fully described below in conjunction with the flowchart of FIG. 3. Basically, the trace interrupt handler will read the addresses in registers 105 and 107, along with the values in registers 109, 111, 113 and determine whether this information relates to tracing or profiling of the instructions corresponding to program 100 that is being run. For profile information, the appropriate register contents are placed in a profile buffer 117. Trace information is then stored in trace buffer 119. These buffers 117 and 119 are then accessible by a user by the normal system I/O, i.e. the results can be viewed on display 38, or stored to disk 20, or the like.

FIGS. 3A–3D are a flowchart illustrating the operation of the trace interrupt handler software of the present invention. This software is part of the operating system, in a preferred embodiment the AIX operating system is contemplated to be used by the present invention. Therefore, interrupt handler 115 is included as part of the AIX operating system.

At step 1 the interrupt handler program is invoked and the process identifies registers 105, 107, 109, 111, 113 at step 2. It is then determined at step 3 whether the

	Document ID	U	Title	Current OR
61	US 5506975 A	☒	Virtual machine I/O interrupt control method compares number of pending I/O interrupt conditions for non-running virtual machines with predetermined number	709/1
62	US 5490259 A	☒	Logical-to-real address translation based on selective use of first and second TLBs	711/202
63	US 5481744 A	☒	Microcode sequencer changing states in response to an external gating input level change upon the occurrence of a wait instruction	712/227
64	US 5465361 A	☒	Microcode linker/loader that generates microcode sequences for MRI sequencer by modifying previously generated microcode sequences	717/10
65	US 5457715 A	☒	Method and device for communicating data using a plurality of lines	375/260
66	US 5442604 A	☒	Access control device	369/44.11
67	US 5428732 A	☒	Playlist mechanism for specification of complex memory objects	345/302
68	US 5408443 A	☒	Programmable medication dispensing system	368/10
69	US 5398319 A	☒	Microprocessor having apparatus for dynamically controlling a kind of operation to be performed by instructions to be executed	712/226
70	US 5386565 A	☒	Method and system for controlling/monitoring computer system having plural operating systems to run thereon	717/4
71	US 5361356 A	☒	Storage isolation with subspace-group facility	711/206
72	US 5349667 A	☒	Interrupt control system for microprocessor for handling a plurality of maskable interrupt requests	710/267
73	US 5333280 A	☒	Parallel pipelined instruction processing system for very long instruction word	712/241
74	US 5321032 A	☒	Peptide compounds and pharmaceutical compositions thereof	514/308
75	US 5291610 A	☒	Microcode sequencer changing states in response to an external gating input level change upon the occurrence of a WAIT instruction	712/226
76	US 5287548 A	☒	Programmable controller having a stored program with both machine language instructions and source code data	700/18
77	US 5283891 A	☒	Error information saving apparatus of computer	714/45
78	US 5280617 A	☒	Automatic program code generation in a compiler system for an instantiation of a generic program structure and based on formal parameters and characteristics of actual parameters	717/3
79	US 5274815 A	☒	Dynamic instruction modifying controller and operation method	712/226
80	US 5270495 A	☒	Combination weighing machine with feed control	177/25.18

current instruction address in register 105 is the same as the last, or previous, instruction address plus one. If not, then the routine proceeds to step 4, since the instructions are not sequential. This means that a branch or system interrupt, or the like, has occurred. Step 5 then determines whether the instruction address information is to be used for tracing or profile. If the information is to be used for profiling, then the count for last instruction address is updated in the profile buffer 117 (step 6). However, for tracing the last instruction address itself is provided to trace buffer 119. Subsequent to both steps 6 and 7, the process continues to step 8. Also, if it was determined that the instructions were sequential, at step 3, the method proceeds to step 8. It is then determined if the instruction was either a load or a store (step 8). If so, then step 9 determines whether the information is to be used for profile or trace. If the instruction was neither a load nor store, the routine skips to step 14 (discussed below). If the load/store determination information is to be used as profile data, then the count information corresponding to the data address, for data being operated on by the instruction, is logged in profile buffer 117, at step 13.

However, for trace information, the actual address of the memory location at which the data that was operated on by the instructions is logged in the trace buffer 119. Subsequent to step 10, it is then determined if the number of bytes moved is equal to 0 (step 11), if not, then the number of bytes moved by the load or store operation are placed in trace buffer 119 (step 12). If the number of bytes moved is equal to 0, i.e. no bytes were moved, then the process continues to step 14. It is then determined if an address mapping change has occurred. More specifically, if register 113 contains a binary "1" then an address mapping change has occurred, i.e. the last instruction changed the effective to virtual address mapping. If a logical "0" is present in register 113, then the last instruction did not change address mapping. As noted above, an address mapping change is an alteration to the look-up table used to translate an effective address to a virtual address. If it is determined that an address mapping change has occurred, then the new mapping (relationship of effective address to virtual address) is stored in trace buffer 119 step 15).

If it was determined that the address mapping did not change, and subsequent to step 15, the current instruction address is saved, at step 16, for comparison purposes. Step 17 sets the last instruction address equal to the current instruction address effectively incrementing the current instruction address to make it the "new" last instruction address. Step 18 then determines if there are more instructions to process. If so, the procedure returns to step 3. If not, the process continues to step 19 and ends.

Following is the pseudocode for the trace interrupt software of the present invention. This pseudocode describes the activity that would take place in the trace interrupt handler, when a trace interrupt is incurred. Note that the pseudocode assumes the existence of five registers, including the instruction address and data address registers. These registers are illustrated in FIG. 2, which is a block diagram of the hardware aspect of the present invention. Along with the instruction and data address registers, a Load_Store register is provided that contains a 1 if the last instruction was a load or store, otherwise it contains 0. The Number_of_Bytes_Moved register contains the number of bytes moved by the last instruction, if the last instruction was a load or store whose operand size is only known at run-time, otherwise it contains 0. The problem here is that standard RISC systems have load and store instructions whose operand size (i.e. number of bytes moved) is given by a register value at run-time: The Address_Mapping_Change register contains a 1 if the last instruction changed the effective-to-virtual address mapping, otherwise, it contains 0.

```
Pseudocode
Registers:
    Instruction_Address
    Data Address
    Load_Store
    Number_of_Bytes_Moved
    Address_Mapping_Change
Variables:
    Last_Instruction_Address (Address of the previous, or
                              last instruction executed)
Code:
    if (Instruction_Address) does not equal
        (Last_Instruction_Address + 1)
    /*  instructions were not sequential, branch or
        interrupt took place */
{
    if (Tracing)
        /* Record non_sequentiality */
        Place Last_Instruction_Address in trace buffer
}
if (Profiling)
    Update Count for Last_Instruction_Address
if (Load_Store equals 1)
    {
    if (Tracing)
        {
        Place Data_Address in trace buffer
        if (Number_of_Bytes_Moved does not equal 0)
            Place Number_of_Bytes in trace buffer
        }
    if (Profiling)
        Update Count for Data Address
    }
if (Address_Mapping_Change equals 1)
    {
    Place new address mapping in trace buffer
    }
/*  save away Instruction_address for comparison next
    time through */
Last_Instruction_Address = Instruction_Address;
Resume execution at instruction at
Instruction_Address;
```

Although certain preferred embodiments have been shown and described it should be understood that many changes and modifications can be made therein without departing from the scope of the appended claims.

We claim:

1. A method of recording trace information relating to the execution of instructions on a processor unit, comprising the steps of:

storing instruction information output from at least one execution unit in said processing unit;

determining from said instruction information whether said instructions access data from a memory subsystem; and

determining from said instruction information, stored in said processor unit, a data address at which said data, operated on by said instructions, was accessed.

2. A method according to claim 1 wherein said step of determining whether said instructions access data, comprises the step of determining a quantity of data accessed from said memory location, and for storing said quantity as trace information.

	Document	U	Title	Current OR
81	US 5258926 A	☒	Method of measuring radiation for a radiation measuring device	250/375
82	US 5251294 A	☒	Accessing, assembling, and using bodies of information	707/512
83	US 5247627 A	☒	Digital signal processor with conditional branch decision unit and storage of conditional branch decision results	712/236
84	US 5237667 A	☒	Digital signal processor system having host processor for writing instructions into internal processor memory	712/248
85	US 5233653 A	☒	Apparatus and method for enciphered facsimile transmission and reception	380/243
86	US 5222241 A	☒	Digital signal processor having duplex working registers for switching to standby state during interrupt processing	712/228
87	US 5206940 A	☒	Address control and generating system for digital signal-processor	711/218
88	US 5202887 A	☒	Access control method for shared duplex direct access storage device and computer system therefor	714/8
89	US 5201039 A	☒	Multiple address-space data processor with addressable register and context switching	711/201
90	US 5182811 A	☒	Exception, interrupt, and trap handling apparatus which fetches addressing and context data using a single instruction following an interrupt	710/264
91	US 5162988 A	☒	Multiplexing character processor	710/105
92	US 5146403 A	☒	Change of address system and method of using same	707/102
93	US 5144242 A	☒	Continually loadable microcode store for MRI control sequencers	324/312
94	US 5143210 A	☒	Foldable protective package for storing, dispensing, and displaying diagnostic kit components	206/738
95	US 5115500 A	☒	Plural incompatible instruction format decode method and apparatus	712/209
96	US 5109335 A	☒	Buffer memory control apparatus using address translation	711/3
97	US 5099718 A	☒	Automatic transmission and speed change control method	477/121
98	US 5088031 A	☒	Virtual machine file control system which translates block numbers into virtual addresses then into real addresses for accessing main storage	709/100
99	US 5050065 A	☒	Reconfigurable multiprocessor machine for signal processing	702/57
100	US 5048870 A	☒	Multipart flag label for pharmaceutical products	283/81
101	US 5046609 A	☒	Kit for distributing pharmaceutical products	206/232

K

HPS Trailer Page
for
Walk-Up_Printing

Printer: cpk2_2b01_gbkaptr

Summary

Document	Pages	Printed	Missed
US005978937	55	55	0
US005964893	30	30	0
US005884023	25	25	0
US005630102	19	19	0
US005625785	14	14	0
US005564028	29	29	0
US005446876	13	13	0
Total (7)	185	185	0

	Document	U	Title	Current OR
102	US 5045993 A	☒	Digital signal processor	712/236
103	US 5032980 A	☒	Information processing system with instruction address saving function corresponding to priority levels of interruption information	712/228
104	US 5029069 A	☒	Data processor	712/234
105	US 5008807 A	☒	Data processing apparatus with abbreviated jump field	712/213
106	US 4999795 A	☒	Portable keyboard operated alpha computer system with few keys and visual keystroke instructions	708/146
107	US 4980845 A	☒	Digital engine analyzer	701/33
108	US 4976351 A	☒	Kit for distributing pharmaceutical products	206/232
109	US 4964046 A	☒	Harvard architecture microprocessor with arithmetic operations and control tasks for data transfer handled simultaneously	712/36
110	US 4958275 A	☒	Instruction decoder for a variable byte processor	712/225
111	US 4924431 A	☒	Keyboard located indicia for instructing a multi-mode programmable computer having alphanumeric capabilities from a few keyboard keys	708/146
112	US 4908612 A	☒	Computer input-output device	345/159
113	US 4904605 A	☒	Method and apparatus for residential water test kit	436/169
114	US 4900033 A	☒	Spelling game apparatus	273/249
115	US 4896258 A	☒	Data processor provided with instructions which refer to both tagged and tagless data	712/236
116	US 4894857 A	☒	Method and apparatus for customer account servicing	379/67.1
117	US 4879646 A	☒	Data processing system with a pipelined structure for editing trace memory contents and tracing operations during system debugging	712/227
118	US 4866598 A	☒	Communications base microcontroller	710/45
119	US 4862347 A	☒	System for simulating memory arrays in a logic simulation machine	703/14
120	US 4860234 A	☒	Portable keyboard operated alpha computer system with few keys and visual keystroke instructions	708/146
121	US 4851833 A	☒	Digital engine analyzer	345/10
122	US 4849744 A	☒	Digital engine analyzer	345/10
123	US 4844469 A	☒	Golf trainer for calculating ball carry	473/225

US005640578A

United States Patent [19]

Balmer et al.

[11] Patent Number: 5,640,578

[45] Date of Patent: Jun. 17, 1997

[54] **ARITHMETIC LOGIC UNIT HAVING PLURAL INDEPENDENT SECTIONS AND REGISTER STORING RESULTANT INDICATOR BIT FROM EVERY SECTION**

[75] Inventors: **Keith Balmer**, Bedford; **Nicholas Ing-Simmons**, Huntingdon, both of England; **Karl M. Guttag**, Missouri City, Tex.; **Robert J. Gove**, Plano, Tex.; **Jeremiah E. Golston**, Sugar Land, Tex.; **Christopher J. Read**, Houston, Tex.; **Sydney W. Poland**, Katy, Tex.

[73] Assignee: **Texas Instruments Incorporated**, Dallas, Tex.

[21] Appl. No.: **158,742**

[22] Filed: **Nov. 30, 1993**

[51] **Int. Cl.**[6] **G06F 7/38**; G06F 7/50

[52] **U.S. Cl.** **395/562**; 364/715.08; 364/716; 364/736.5; 364/768

[58] **Field of Search** 395/375, 775; 364/715.08, 736.5, 748, 744, 716, 768

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,937,940	2/1976	Brantingham	235/156
4,179,746	12/1979	Tubbs	364/900
4,312,034	1/1982	Gunter et al.	364/200
4,451,885	5/1984	Gerson et al.	364/200
4,467,444	8/1984	Harmon, Jr. et al.	364/900
4,550,437	10/1985	Kobayashi et al.	382/41
4,592,005	5/1986	Kregness	364/736
4,653,075	3/1987	Wisniewski	375/110
4,689,807	8/1987	Mann	377/57
4,692,888	9/1987	New	364/728
4,752,893	6/1988	Guttag et al.	364/518
4,785,393	11/1988	Chu et al.	364/200
4,789,957	12/1988	Niehaus et al.	364/749
4,811,266	3/1989	Woods et al.	364/736.5
4,847,802	7/1989	Ashton	364/748
4,872,131	10/1989	Knbota et al.	364/736
4,888,722	12/1989	Boreland	364/736.5

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

2228652 10/1993 United Kingdom .

OTHER PUBLICATIONS

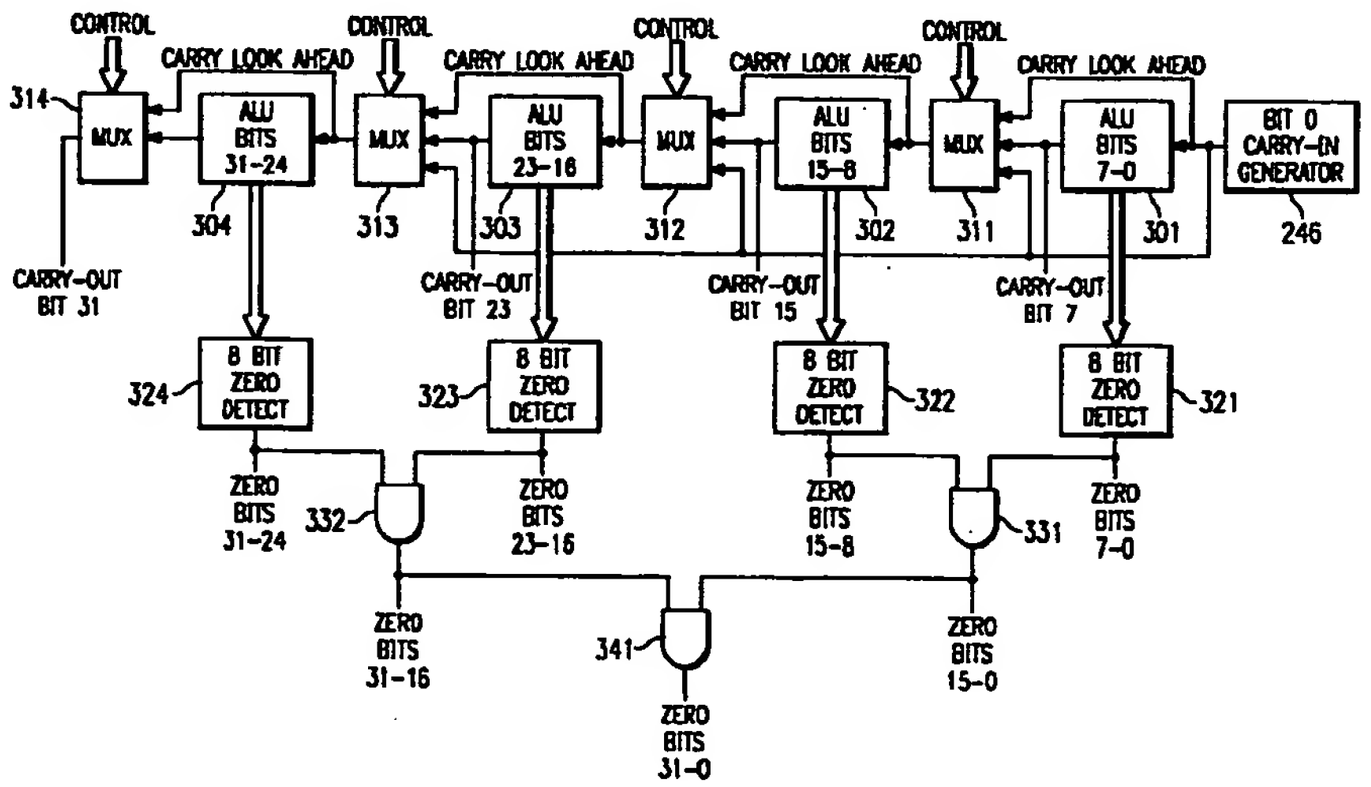
Microprocessor Report—Slater, Michael, "TI Ships Programmable Video Processor", vol. 5, No. 20, Oct. 30, 1991 pp. 1, 6–7, 13.

Primary Examiner—Thomas C. Lee
Assistant Examiner—Rehana Perveen Krick
Attorney, Agent, or Firm—Robert D. Marshall, Jr.; James C. Kesterson; Richard L. Donaldson

[57] **ABSTRACT**

An arithmetic logic unit (**230**) may be divided into a plurality of independent sections (**301, 302, 303, 340**). A bit zero of carry status signal corresponding to each section that is stored in a flags register (**211**), which preferably includes more bits than the maximum number of sections of the arithmetic logic unit (**230**). New status signals may overwrite the previous status signals or rotate the stored bits and store the new status signals. A status register (**210**) stores a size indicator that determines the a number of sections of the arithmetic logic unit (**230**). A status detector has a zero detector (**321, 322, 323, 324**) for each elementary section (**301, 302, 303, 304**) of the arithmetic logic unit (**230**). When there are fewer than the maximum number of sections, these zero signals are ANDed (**331, 332, 341**). A multiplexer couples the carry-out of an elementary (**311, 312, 313, 314**) to the carry-in of an adjacent elementary section (**301, 302, 303, 304**) or not depending on the selected number of sections. The status detector supplies carry outs from each elementary section (**301, 302, 303, 304**) not coupled to an adjacent elementary section (**301, 302, 303, 304**) to the flags register (**211**). Status signals stored in the flags register (**211**) influence the combination of inputs formed by the arithmetic logic unit (**230**) within corresponding sections. An expand circuit (**238**) expands selected bits of flags register (**211**) to form a third input to a three input arithmetic logic unit (**230**).

77 Claims, 37 Drawing Sheets



	Document	U	Title	Current OR
124	USD 48398 90 A	☒	Data bit synchronizer	370/503
125	US 48274 01 A	☒	Method and apparatus for synchronizing clocks prior to the execution of a flush operation	713/400
126	US 48129 74 A	☒	Data processing apparatus for processing list vector instructions	711/218
127	US 48127 68 A	☒	Digital engine analyzer	324/379
128	US 48049 21 A	☒	Digital engine analyzer	324/394
129	US 48036 20 A	☒	Multi-processor system responsive to pause and pause clearing instructions for instruction execution control	712/203
130	US 48021 65 A	☒	Method and apparatus of debugging computer programs	714/38
131	US 48003 78 A	☒	Digital engine analyzer	345/10
132	US 47991 46 A	☒	System for displaying graphic information on video screen employing video display processor	710/260
133	US 47808 10 A	☒	Data processor with associative memory storing vector elements for vector conversion	712/6
134	US 47681 57 A	☒	Video image processing system	345/517
135	US 47589 49 A	☒	Information processing apparatus	712/208
136	US 47181 05 A	☒	Graphic vectorization system	382/243
137	US 47137 51 A	☒	Masking commands for a second processor when a first processor requires a flushing operation in a multiprocessor system	712/203
138	US 46358 34 A	☒	Apparatus for pattern crocheting	223/107
139	US 46302 88 A	☒	Data transmission with block coding	375/261
140	US 46213 39 A	☒	SIMD machine using cube connected cycles network architecture for vector processing	712/22
141	US 46111 13 A	☒	Heat stress calculator	235/78R
142	US 45919 67 A	☒	Distributed drum emulating programmable controller system	700/3
143	US 45439 57 A	☒	Human response apparatus and method	600/300
144	US 45325 89 A	☒	Digital data processor with two operation units	712/217
145	US 44882 57 A	☒	Method for confirming incorporation of a memory into microcomputer system	711/102

U.S. PATENT DOCUMENTS

4,901,270	2/1990	Galbi et al.	364/786
4,924,422	5/1990	Vassiliadis et al.	364/715.09
4,933,878	6/1990	Guttag et al.	364/521
4,985,848	1/1991	Pfeiffer et al.	364/518
5,020,014	5/1991	Maher, III et al.	364/715.08
5,081,698	1/1992	Kohn	395/122
5,095,301	3/1992	Guttag et al.	340/703
5,103,419	4/1992	Toyokura et al.	364/750.5
5,140,687	8/1992	Dye et al.	395/500
5,185,714	2/1993	Nakayama	364/750.5
5,197,140	3/1993	Balmer	395/400
5,206,828	4/1993	Shah et al.	364/784
5,212,777	5/1993	Gove et al.	395/375
5,226,125	7/1993	Balmer et al.	395/325
5,231,694	7/1993	Novak et al.	395/162
5,239,654	8/1993	Ing-Simmons et al.	395/800
5,249,266	9/1993	Dye et al.	395/162
5,294,918	3/1994	Preston et al.	345/155

	Document	U	Title	Current OR
146	USD 44563 49 A	☒	Cinematographic camera	352/91C
147	US 44498 05 A	☒	Data registration device	396/318
148	US 44438 49 A	☒	Error recovery system of a multi-processor system for recovering an error by transferring status singals from one processor to another without use of a main memory	714/10
149	US 44098 44 A	☒	Flow rate measuring device	73/861
150	US 44096 21 A	☒	Method and apparatus for compacting and decompacting character in accordance with a variety of methods	358/430
151	US 44007 73 A	☒	Independent handling of I/O interrupt requests and associated status information transfers	710/19
152	US 43847 71 A	☒	Data registration device	396/318
153	US 43171 71 A	☒	LSI Microprocessor having an error processing circuit	712/244
154	US 43143 30 A	☒	Machine tool data system	700/87
155	US 43108 61 A	☒	Data-recording device	360/50
156	US 43090 89 A	☒	Exposure indicating apparatus responsive to film exposure latitude	396/208
157	US 42770 89 A	☒	Pharmaceutical record and label system	462/67
158	US 42589 22 A	☒	Computer math game	273/302
159	US 42412 38 A	☒	Telephone number memory and indicator system	379/142
160	US 42321 99 A	☒	Special services add-on for dial pulse activated telephone switching office	379/197
161	US 42109 41 A	☒	Data-recording device	360/48
162	US 42086 57 A	☒	Electronic automatic plotter	342/182
163	US 42003 69 A	☒	Method and apparatus for compacting and decompacting character data in accordance with a variety of methods	396/551
164	US 41975 78 A	☒	Microprogram controlled data processing system	712/211
165	US 41504 27 A	☒	Machine tool data system and method	700/180
166	US 41060 90 A	☒	Monolithic microcomputer central processor	712/32

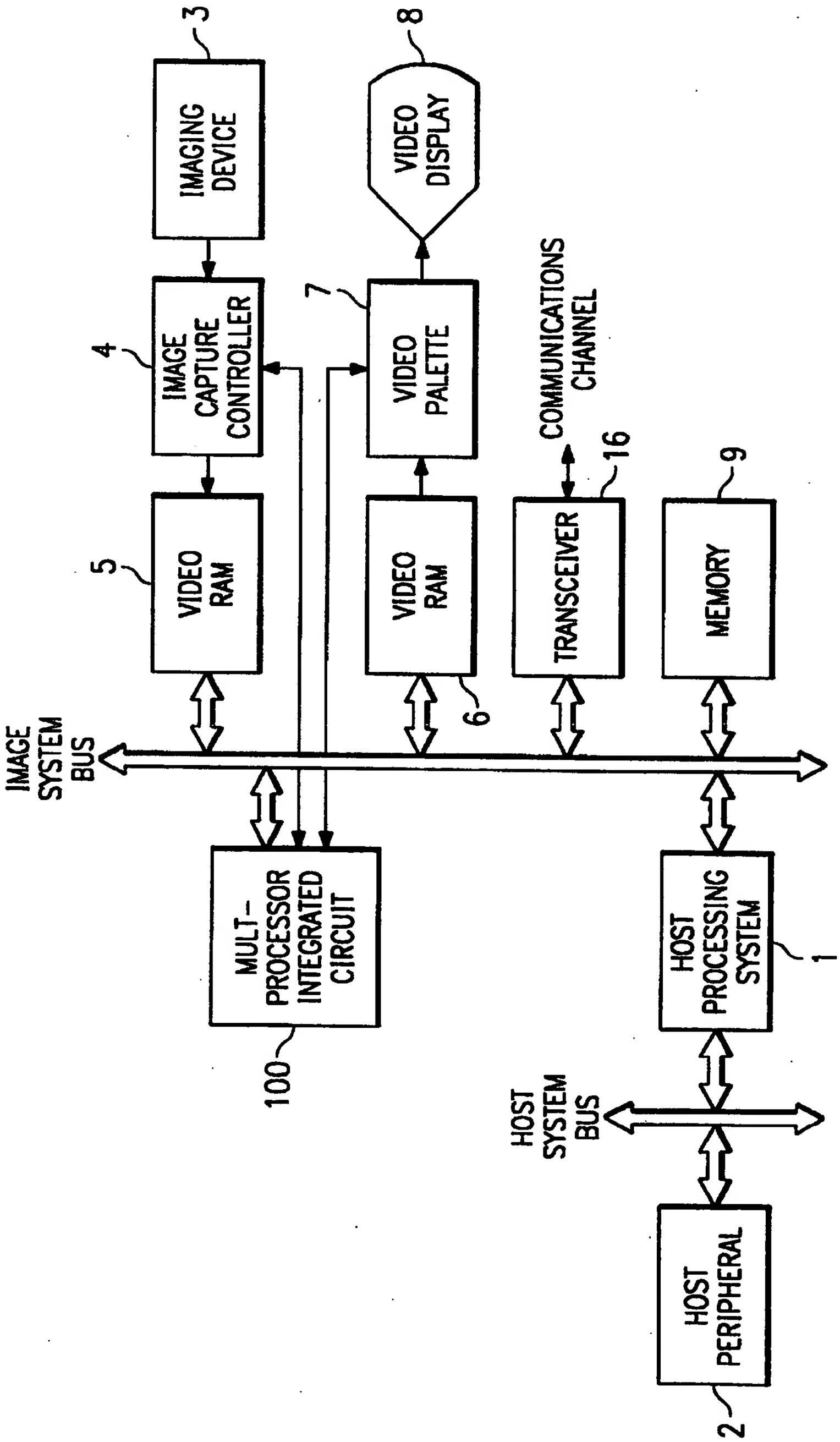

IMAGING DEVICE
3
VIDEO DISPLAY
8
IMAGE CAPTURE CONTROLLER
4
VIDEO PALETTE
7
COMMUNICATIONS CHANNEL
VIDEO RAM
5
VIDEO RAM
6
TRANSCEIVER
16
MEMORY
9
IMAGE SYSTEM BUS
MULT-PROCESSOR INTEGRATED CIRCUIT
100
HOST PROCESSING SYSTEM
1
HOST SYSTEM BUS
HOST PERIPHERAL
2

FIG. 1

	Document	U	Title	Current OR
167	US 41047 21 A	☒	Hierarchical security mechanism for dynamically assigning security levels to object programs	711/164
168	US 40927 19 A	☒	Autoadaptive working center for programmable automation	700/95
169	US 40878 52 A	☒	Microprocessor for an automatic word-processing system	712/220
170	US 40740 67 A	☒	Digital printout arrangement with magnetic field carriage drive	178/23R
171	US 40588 50 A	☒	Programmable controller	711/117
172	US 39361 82 A	☒	Control arrangement for an electrostatographic reproduction apparatus	399/77
173	US 38817 32 A	☒	RING, POST AND LOOP PUZZLE WITH INDIVIDUAL POST ANCHORS	273/158
174	US 38167 23 A	☒	MACHINE TOOL DATA SYSTEM AND METHOD	318/591
175	US 38124 75 A	☒	DATA SYNCHRONIZER	710/7
176	US 37878 91 A	☒	SIGNAL PROCESSOR INSTRUCTION FOR NON-BLOCKING COMMUNICATION BETWEEN DATA PROCESSING UNITS	709/229
177	US 37787 80 A	☒	OPERATION REQUEST BLOCK USAGE	709/102
178	US 37741 66 A	☒	SHORT-RANGE DATA PROCESSING TRANSFERS	711/200
179	US 37256 52 A	☒	COMPUTER CONTROLLED MACHINE TOOL SYSTEM WITH STORED MACRO LANGUAGE PROGRAM FOR EFFECTING PATTERN TYPE PUNCHING	318/568.1
180	US 37178 50 A	☒	PROGRAMMED DATA PROCESSING WITH FACILITATED TRANSFERS	712/205
181	US 36987 19 A	☒	RING PUZZLE WITH QUICK RESTORATION MEANS	273/158
182	US 36817 61 A	☒	ELECTRONIC DATA PROCESSING SYSTEM WITH PLURAL INDEPENDENT CONTROL UNITS	712/230
183	US 36264 27 A	☒	LARGE-SCALE DATA PROCESSING SYSTEM	712/244
184	US 36102 00 A	☒	PLACE-MARKING DEVICE	116/240
185	US 35949 20 A	☒	DRIVING INSTRUCTION FOR PERSONS WITH A HEARING HANDICAP	434/66
186	US 35917 66 A	☒	PREDICTED ITERATION IN DECIMAL DIVISION	708/652
187	US 35545 48 A	☒	FOOTBALL GAME	273/247

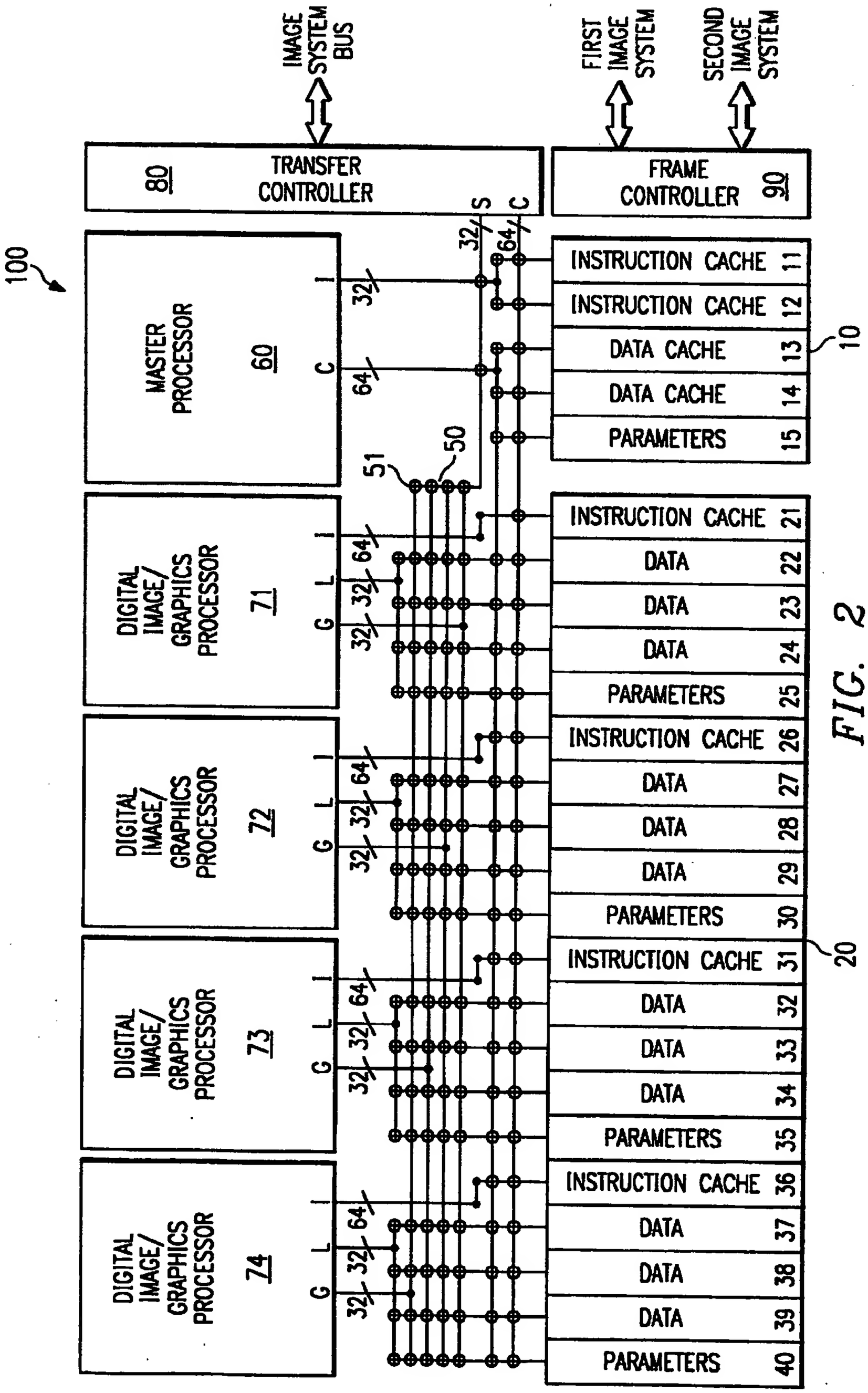


IMAGE SYSTEM BUS
FIRST IMAGE SYSTEM
SECOND IMAGE SYSTEM
80 TRANSFER CONTROLLER
FRAME CONTROLLER 90
100
MASTER PROCESSOR 60
DIGITAL IMAGE/ GRAPHICS PROCESSOR 71
DIGITAL IMAGE/ GRAPHICS PROCESSOR 72
DIGITAL IMAGE/ GRAPHICS PROCESSOR 73
DIGITAL IMAGE/ GRAPHICS PROCESSOR 74
I
C
G L
S
32
64
50
51
INSTRUCTION CACHE 11
INSTRUCTION CACHE 12
DATA CACHE 13
DATA CACHE 14
PARAMETERS 15
10
INSTRUCTION CACHE 21
DATA 22
DATA 23
DATA 24
PARAMETERS 25
INSTRUCTION CACHE 26
DATA 27
DATA 28
DATA 29
PARAMETERS 30
INSTRUCTION CACHE 31
DATA 32
DATA 33
DATA 34
PARAMETERS 35
INSTRUCTION CACHE 36
DATA 37
DATA 38
DATA 39
PARAMETERS 40
20

FIG. 2